

Northern Michigan University

NMU Commons

---

All NMU Master's Theses

Student Works

---

7-2021

## Computable Model Theory on Loops

Josiah Schmidt  
joschmid@nmu.edu

Follow this and additional works at: <https://commons.nmu.edu/theses>



Part of the [Algebra Commons](#), and the [Other Mathematics Commons](#)

---

### Recommended Citation

Schmidt, Josiah, "Computable Model Theory on Loops" (2021). *All NMU Master's Theses*. 682.  
<https://commons.nmu.edu/theses/682>

This Open Access is brought to you for free and open access by the Student Works at NMU Commons. It has been accepted for inclusion in All NMU Master's Theses by an authorized administrator of NMU Commons. For more information, please contact [kmcdonou@nmu.edu](mailto:kmcdonou@nmu.edu), [bsarjean@nmu.edu](mailto:bsarjean@nmu.edu).

Computable Model Theory on Loops

By

Josiah Schmidt

THESIS

Submitted to

Northern Michigan University

In partial fulfillment of the requirements

For the degree of

MASTER OF SCIENCE

Office of Graduate Education and Research

July 2021

© 2021 Northern Michigan University

SIGNATURE APPROVAL FORM

COMPUTABLE MODEL THEORY ON LOOPS

This thesis by Josiah Schmidt is recommended for approval by the student's Thesis Committee, the Department Head of the Department of Mathematics and Computer Science, and the Dean of Graduate Education and Research.



Date 7/9/2021

Committee Chair: Joshua Thompson, Associate Professor



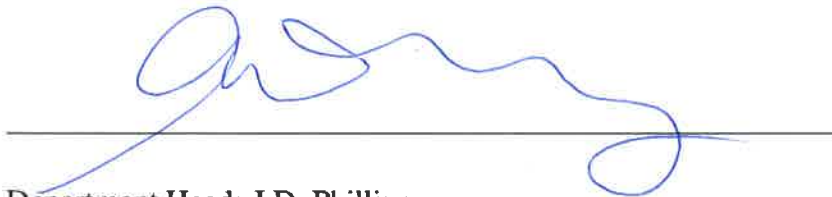
Date 7/9/2021

First Reader: Linda Lawton, Associate Professor



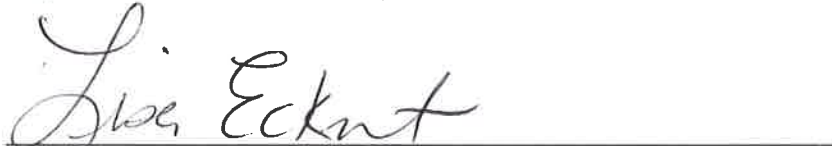
Date 7/9/2021

Second Reader: J.D. Phillips, Professor



Date 7/9/2021

Department Head: J.D. Phillips



Date 8/12/2021

Dean of Graduate Education and Research: Dr. Lisa Eckert



## ABSTRACT

### Computable Model Theory on Loops

By

Josiah Schmidt

We give an introduction to the problem of computable algebras. Specifically, the algebras of loops and groups. We start by defining a loop and group, then give some of their properties. We then give an overview of computability theory, and apply it to loops and groups. We conclude by showing that a finitely presented residually finite algebra has a solvable word problem.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Loops . . . . .	1
1.2	Computability Problems in Loops . . . . .	2
<b>2</b>	<b>Loops and Groups</b>	<b>5</b>
	Example 2.1 . . . . .	9
2.1	Loops . . . . .	9
	Example 2.2 . . . . .	10
	Example 2.3 . . . . .	12
2.1.1	Inverses . . . . .	12
	Example 2.4 . . . . .	14
	Example 2.5 . . . . .	16
	Example 2.6 . . . . .	16
2.1.2	subloops . . . . .	17
2.1.3	Moufang Loops . . . . .	21
	Example 2.7 . . . . .	23
2.2	Groups . . . . .	25
2.2.1	Subgroups . . . . .	28
<b>3</b>	<b>Computability Theory</b>	<b>31</b>
3.1	Formal Definitions of Computable Functions . . . . .	31
3.1.1	Turing Machines . . . . .	31
3.1.2	Primitive Recursive Functions . . . . .	34
	Example 3.1 . . . . .	35
3.2	Computability Background . . . . .	37
3.2.1	Basic Results . . . . .	37
	Example 3.2 . . . . .	38
	Example 3.3 . . . . .	38

3.2.2	Recursively Enumerable Sets . . . . .	40
<b>4</b>	<b>Applied Computability</b>	<b>42</b>
4.1	Computable Groups . . . . .	42
4.2	Computable Loops . . . . .	44
4.3	Finitely Presented Algebras . . . . .	48
	Example 4.1 . . . . .	49
	<b>References</b>	<b>52</b>

# List of Figures

1	Turing Machine . . . . .	32
---	--------------------------	----



# 1 Introduction

## 1.1 Loops

The Main goal of this thesis is to give an overview on the topic of computable algebras. We focus mainly on two specific algebraic structures, loops and groups. We start by giving the simplest algebra to define, the groupoid. A groupoid is a set  $G$ , along with a binary (closed) operation, called multiplication (product), denoted as  $\cdot$ . A groupoid  $G$  is denoted as  $(G, \cdot)$ . If we consider a fixed element in  $G$ , say  $a$ , we have that  $a$  can either be multiplied on the right hand side, or the left hand side by an arbitrary element, say  $x$ , in  $G$ . We let  $xR(a) = a \cdot x$  and  $xL(a) = x \cdot a$  denote these two possible choices. Both  $R(a)$ , and  $L(a)$  are translation mappings of the groupoid  $G$ , and are used to define a quasigroup.

We define a quasigroups as a groupoid  $(G, \cdot)$  where both of its translation mappings are bijections. A mapping is a bijection if it satisfies two conditions (1) if for every  $u' \in B$  there exists  $u \in A$  such that  $u' = u\alpha$  (surjective) (2) whenever  $u = v$  where  $u, v \in A$ , we have that  $u\alpha = v\alpha$  (injective). Given the bijectivity of these translation maps, we are able to define two inverses mappings  $R(a)^{-1}$  and  $L(a)^{-1}$ . These mappings are not necessarily translation mappings, and are denoted as  $/$  and  $\backslash$  respectively. These mappings are defined as  $x \backslash y = yL(x)^{-1}$  and  $x/y = xR(y)^{-1}$  for all  $x, y \in G$ . With these three operations,  $\cdot$ ,  $/$ , and  $\backslash$ , we are able to define a loop.

A loop can be define as a set  $G$  together with three binary operations  $(\cdot)(\backslash)(/)$ , denoted as  $(G, \cdot, \backslash, /)$ , such that (1)  $a \cdot (a \backslash b) = b$ ,  $(b/a) \cdot a = b$  for all  $a, b \in G$ , (2)  $a \backslash (a \cdot b) = b$ ,  $(b \cdot a)/a = b$  for all  $a, b \in G$ , (3)  $a \backslash a = b/b$  for all  $a, b \in G$ . Alternatively, a loop can be defined as a quasigroup with an identity element. An identity element is an element  $e$  inside a set  $G$  that satisfies the conditions of the  $L(e) : G \rightarrow G$  and  $R(e) : G \rightarrow G$  mappings.

We go onto look at some special types of loops with varying properties. The inverse property loop is defined as a loops satisfying the equations  $a^\lambda \cdot (a \cdot x) = x$  and  $(x \cdot a) \cdot a^\rho = x$ . A loop that is Lagrange-like, defined by a finite loop  $G$  of order  $n$  having a subloop (A non-empty subset of a loop that is also loop)  $H$  of order  $m$  such that divides  $m|n$ . A commutative loop is a loop where the translation maps,  $L(a)$  and  $R(a)$ , satisfy  $L(a) = R(a)$  for all  $a \in G$ . And lastly, an associative loop is a loop where the translation maps,  $L(a)$  and  $R(a)$ , satisfy  $R(a \cdot b) = R(a)R(b)$  for all  $a, b \in G$ .

We call an associative loop a group. The reason is due to the additional structure a loop acquires when this property is assumed. All associative loops satisfy the inverse property, as well as a stronger form of the Lagrange-like property, known as the strong Lagrange-like property. These are the most researched loop structures. The second most commonly studied loop was introduced by the German mathematician Ruth Moufang. This loop structure, known as a Moufang loop, satisfies a property that is very similar to the associative property, called the Moufang identity. We show these two loops to have solvable word problem in the finitely generated case.

## 1.2 Computability Problems in Loops

Computability theory had its early beginnings in the early 1930's when Gödel gave his famous Incompleteness Theorem. To prove this theorem the notion of a primitive recursive function was given. This notion led Church, Kleene, Post, Turing, and Gödel himself to define what a recursive function was. These definitions, all varying from one another, were proved to all give rise to exactly the same class of mathematical functions. We can think of these functions as being a computer program which has infinite storage capacity, that gives an answer after a finite amount

of time.

A similar notion to computability is the notion of computably listable set of numbers. Said another way, a set which can be generated by a computable procedure. We examine these computably listable sets (which we call computably enumerable) and define the interesting set  $K$ .

The word problem for groups was a question proposed by Dehn in 1911. The question came about while he was studying the fundamental groups of manifolds. The question asked is simply stated, yet the proof was something never seen before that time. Let  $G$  be a group generated by a finite set of elements  $a, b, c, \dots$ , and relations  $r(a, b, c, \dots)$ . Is there a uniform test or algorithm for deciding whether an arbitrary word  $w(a, b, c, \dots) = 1$  in  $G$ . The proof of this question did not come until 44 years later. It was independently proven by both Boone in 1959 and Novicov in 1955 that the word problem for finitely generated groups is unsolvable. This proof could not have come about without the works of Marcov, Church, Post, Turing, Kleene, and many others who laid the foundations of what we now call computability theory. Without their precise definition of an algorithm, one could not prove a problem unsolvable.

The word problem, however, is not a question unique only to groups. A similar problem was posed by Thue dealing with an "abstract language"  $L$ . A language  $L$  has a finite alphabet  $a, b, c, \dots$ , a finite list of words composed by that alphabet, and a dictionary. The dictionary is a listing of a finite set of pairs of words. If a word  $w$  is of the form  $uvt$ , where  $u, v$ , are words with  $t$  being a word paired with the word  $s$  in the dictionary, then the word  $uvt$  can be transformed into the word  $uvs$ . If there is a finite sequence of transformations connecting two words  $w$ , and  $w'$ , then we say they are equivalent in  $L$ . This new system gave rise to the word problem for the Language  $L$ , where the question became whether an algorithm existed to determine if two words were equivalent in  $L$ .

The structure theory of algebras naturally give rise to these word problems when the algebra is defined by generators and relations. With this one can ask the question, is the word problem solvable for other algebraic structures? Or, are there other instances where the word problem would be solvable for groups, i.e. not finitely generated groups? We prove one such word problem in the case of all algebras that are finitely presented and residually finite.

The citation method for this thesis is Chicago.

## 2 Loops and Groups

We start this section off by covering some of the rudiments of set theory needed for this paper, as well as the notation which is used when referring to algebraic structures throughout the paper. We then briefly discuss the simple algebra of a groupoid, then work our way through quasigroups to land in loops. We stay long enough in loops to go through some of their structures and substructures to help give a general understanding of them, which will prove beneficial in the later half of the paper. We then head up to one of the most structured algebras, groups, where we also go through some of their structures and substructure.

Let  $A, B$ , and  $C$  be sets. We write  $\alpha : A \rightarrow B$  to mean that  $\alpha$  is a map or function from  $A$  to  $B$  with  $A$  serving as the domain of  $\alpha$ . If we have that  $\alpha : A \rightarrow B$  where  $a \in A$ , we write  $a\alpha$  or  $(a)\alpha$  to denote that element in  $B$  to which  $a$  corresponds under the action or application of  $\alpha$ . If  $\alpha : A \rightarrow B$  and  $\beta : B \rightarrow C$  we define the composite  $\alpha\beta$  by  $(a)\alpha\beta = (a\alpha)\beta$  for all  $a \in A$ . Also, note that the composite  $\alpha\beta : A \rightarrow C$ . We say that a map  $\alpha : A \rightarrow B$  is surjective if for every  $u' \in B$  there exists  $u \in A$  such that  $u' = u\alpha$ . A map  $\alpha : A \rightarrow B$  is injective if, whenever  $u = v$  where  $u, v \in A$ , we have that  $u\alpha = v\alpha$ . A map is bijective whenever it is both surjective and injective. We let  $A \times B$  denote the Cartesian product of  $A$  and  $B$  where we regard the elements of  $A \times B$  as ordered pairs  $(a, b)$  with  $a \in A$  and  $b \in B$ . Also, let  $B^A$  denote the set whose members are those maps from  $A$  to  $B$  which have  $A$  as their domain, and let  $2^A$  be the set of all subsets of  $A$ . The cardinality of a set  $S$  is denoted as  $|S|$ . Whenever  $A$  and  $B$  are finite we have that  $|A \times B| = |A||B|$ ,  $|B^A| = |B|^{|A|}$ , and  $|2^A| = 2^{|A|}$ . We shall denote by  $\iota$  the *identity mapping*  $\iota : x \mapsto x$ . Lastly, when the product notation is in use, we let the juxtaposition  $ab$  stand for  $a \cdot b$ , and  $ab \cdot d$  stand for  $(a \cdot b) \cdot d$ .

A groupoid is a non-empty set  $G$  with a binary (closed) operation. A set  $G$  with

finite order ( $|G| = n$ ) has  $n^{(n^2)}$  possible binary operations in total. This can be seen by considering its power set  $G^{G \times G}$ . Let  $(G, \cdot)$  be a groupoid and let  $a$  be any fixed element in  $G$ . Then the translation maps  $L(a)$  and  $R(a)$  are defined by

$$xL(a) = a \cdot x \text{ and } xR(a) = x \cdot a$$

for all  $x \in G$ . Thus  $L(a) : G \rightarrow G$  and  $R(a) : G \rightarrow G$  for each  $a \in G$ . With these two translation maps  $L(a)$ , and  $R(a)$  of a groupoid, we now define a quasigroup.

**Definition 2.1** *A groupoid  $(G, \cdot)$  is called a quasigroup if the maps  $L(a) : G \rightarrow G$   $R(a) : G \rightarrow G$  are bijections for all  $a \in G$*

From definition 2.1 we see that if the groupoid  $(G, \cdot)$  is a quasigroup, given any two elements in  $G$  the third element is uniquely determined in  $G$ . Also, given that the mappings  $L(a)$  and  $R(a)$  are bijective, quasigroups satisfy the cancellation laws (i.e. for  $a, x, y \in G$ ,  $x \cdot a = y \cdot a$  implies that  $x = y$ , and  $a \cdot x = a \cdot y$  implies that  $x = y$ ).

We now record this observation in the following theorem.

**Theorem 2.1** *Let  $(G, \cdot)$  be a quasigroup. Then for  $(a, b) \in G \times G$  there exists a unique  $(x, y) \in G \times G$  so that  $a \cdot x = y \cdot a = b$  (unique solvability)*

Proof: Assume for contradiction that the ordered pair  $(x, y)$  was not unique. This would mean that there existed another ordered pair  $(r, s) \in G$  such that  $a \cdot r = a \cdot x = s \cdot a = y \cdot a = b$ . However, the translation maps  $R(a)$  and  $L(a)$  of a quasigroup are bijections. Hence, having  $a \cdot r = a \cdot x = b$  implies  $x = r$ , and having  $s \cdot a = y \cdot a = b$  implies that  $y = s$  which is a contradiction of the of the assumption that the ordered pair  $(x, y)$  was not unique. ■

We also make note of the left and right inverse mappings,  $L(a)^{-1}$ , and  $R(a)^{-1}$ , that come about from the bijectivity of  $L(a)$ , and  $R(a)$ . We denote these mappings as  $\backslash$  and  $/$  respectively. We define these mappings as follows.

$$x \backslash y = yL(x)^{-1} \text{ and } x / y = xR(y)^{-1}$$

for all  $x, y \in G$ . Notice that  $x \setminus y = z$  if and only if  $x \cdot z = y$  and that  $x / y = z$  if and only if  $z \cdot y = x$ . Thus  $(G, \setminus)$  and  $(G, /)$  are both quasigroups due to the unique solutions of the quasigroup  $(G, \cdot)$ . These quasigroups  $((G, \setminus)$  and  $(G, /)$ ) are called conjugates of  $(G, \cdot)$ . It is also worth noting that there exist three additional conjugates  $(G, \circ)$ ,  $(G, *)$ , and  $(G, \times)$ ; however, the interest of this paper will only be concerned with the two conjugate  $(G, \setminus)$  and  $(G, /)$ . Also note that the operations  $L(a)^{-1}$  and  $R(a)^{-1}$  are not necessarily translation mappings. This means that for a given  $a \in G$  there is no guarantee, in general, that there is a  $b \in G$  such that  $L(a)^{-1} = L(b)$  or  $L(a)^{-1} = R(b)$ .

We acknowledge that an example of a quasigroup, along with its two conjugates  $(G, \setminus)$  and  $(G, /)$ , would be of great help to our reader in understanding these. However, we find ourselves missing a way to create a finite quasigroup to do so. We remedy this with the following theorem.

**Theorem 2.2** *Let  $(G, \cdot)$  be a finite groupoid. Then the following are equivalent:*

- (i)  $(G, \cdot)$  is a quasigroup
- (ii)  $L(a) : G \rightarrow G$  and  $R(a) : G \rightarrow G$  are injective for all  $a \in G$
- (iii)  $L(a) : G \rightarrow G$  and  $R(a) : G \rightarrow G$  are surjective for all  $a \in G$
- (iv) The right and left cancellation laws hold for  $(G, \cdot)$
- (v) Each element in  $G$  appears once and only once in each row and in each column of a Cayley table  $(G, \cdot)$

Proof:

(i)  $\Rightarrow$  (ii) This follows directly from definition 2.3, since the both right and left mappings of a quasigroup are bijective.

(ii)  $\Rightarrow$  (iii) Given that  $G$  is finite, we have that  $|G| = n$ . Assuming that both translation  $L(a)$  and  $R(a)$  are injective means that the image space of these translation maps contain at most  $n$  elements. However, the image of the translation maps are contained in the finite set  $G$  due to  $G$  being a groupoid. This means that the image of the translation maps contain exactly  $n$  elements, and is therefore surjective.

(iii)  $\Rightarrow$  (iv) Assuming that the translation map  $L(a)$  is surjective means that for each  $y$  in  $G$  there exists an  $x$  in  $G$  such that  $y = xL(a)$ . Given that  $G$  is a finite groupoid, we have that  $|G| = n$ . This implies that for every element  $x$  in  $G$  to cover the image space  $G$  no two left translation mappings can send differing elements to the same image, thus  $L(a)$  is injective. Since  $L(a)$  is injective, the left cancellation property holds. (Prove?). Similarly it can be shown that right cancellation holds.

(iv)  $\Rightarrow$  (v) We shall prove this by showing its contrapositive. Let at least one element appear more than once in a row. This would mean that for two distinct elements  $x, y \in G$   $xL(a) = yL(a) = z$ , thus the left cancellation law does not hold. Now let at least one element appear more than once in each row. This would mean that for two distinct elements  $x, y \in G$   $xR(a) = yR(a) = z$ , thus the right cancellation law does not hold.

(v)  $\Rightarrow$  (i) For every element to appear once and only once in each column and row of a Cayley table means that there exists a unique  $(x, y) \in G \times G$  so that  $a \cdot x = y \cdot a = b$ . Thus, by Theorem 2.1, the Cayley table is a quasigroup. ■

With Theorem 2.2 now in hand, we present the following finite quasigroup along with the conjugates  $(G, \backslash)$  and  $(G, /)$ .



### Example 2.1

	·		1	2	3	4	5		/		1	2	3	4	5		\		1	2	3	4	5
	1		5	3	4	1	2		1		4	5	2	1	3		1		4	5	2	3	1
	2		3	5	1	2	4		2		3	4	5	2	1		2		3	4	1	5	2
	3		2	4	3	5	1		3		2	1	3	5	4		3		5	1	3	2	4
	4		1	2	5	4	3		4		5	3	1	4	2		4		1	2	5	4	3
	5		4	1	2	3	5		5		1	2	4	3	5		5		2	3	4	1	5

## 2.1 Loops

We start this section off by giving the algebraic definition of a loop. From there we go onto to show two areas of interest for loops, their inverses, and substructures. We then go onto defining one of the more structured cases of a loop, called a moufang loop, and end with some of the properties of commutative moufang loops.

**Definition 2.2** *A loop  $(G, \cdot, \backslash, /)$  is a set  $G$  together with three binary operations  $(\cdot)(\backslash)(/)$  such that*

$$(i) \ a \cdot (a \backslash b) = b, (b/a) \cdot a = b \text{ for all } a, b \in G$$

$$(ii) \ a \backslash (a \cdot b) = b, (b \cdot a)/a = b \text{ for all } a, b \in G$$

$$(iii) \ a \backslash a = b/b \text{ for all } a, b \in G$$

Even though the definition of a loop can be given in this short and straight forward manner, it is easy to find oneself struggling with the notation. In addition, dealing with infinite nonassociative objects is not something one deals with a lot of in depth in mathematics. Much of this can be contributed to the alluring structured properties one has when dealing with associative, and symmetric objects. To help

give a better feel for the objects of interest in this paper, we break down this definition by going to the finite setting. We start by giving the following definitions.

**Definition 2.3** Let  $(G, \cdot)$  be a quasigroup and let  $e \in G$ . Then  $e$  is a left (right) identity element for  $(G, \cdot)$  means that  $L(e) : G \rightarrow G$  ( $R(e) : G \rightarrow G$ ) is the left (right) identity map. Also  $e$  is an identity element for  $(G, \cdot)$  means that  $e$  is a left and a right identity element for  $(G, \cdot)$ .

**Definition 2.4** A groupoid  $(G, \cdot)$  is called a loop means that  $(G, \cdot)$  is a quasigroup and  $(G, \cdot)$  has an identity element.

Notice that Definition 2.6 is a reiteration of Definition 2.4 part (iii). We include Definition 2.6 to show that, in the finite case, a Cayley table with an identity element is a loop. We use this to create our finite loop in the following example. In the example we construct a finite loop, and give its right and left inverse mappings.

**Example 2.2**

$\cdot$	1	2	3	4	5	$\setminus$	1	2	3	4	5	$/$	1	2	3	4	5
1	1	2	3	4	5	1	1	2	3	4	5	1	1	5	4	2	3
2	2	3	5	1	4	2	4	1	2	5	3	2	2	1	3	4	5
3	3	4	2	5	1	3	5	3	1	2	4	3	3	2	1	5	4
4	4	5	1	2	3	4	3	4	5	1	2	4	4	3	5	1	2
5	5	1	4	3	2	5	2	5	4	3	1	5	5	4	2	3	1

We now break down Definition 2.4 by going through each part, with a specific instances of the given loop in Example 2.2.

Property (i):  $2 \cdot (2 \setminus 3) = 3 \Rightarrow 2 \cdot 2 = 3 \Rightarrow 3 = 3$ , and  $(5/3) \cdot 3 = 5 \Rightarrow 2 \cdot 3 = 5 \Rightarrow 5 = 5$ .

Property (ii):  $4 \setminus (4 \cdot 2) = 2 \Rightarrow 4 \setminus 5 = 2 \Rightarrow 2 = 2$ , and  $(5 \cdot 4)/4 = 5 \Rightarrow 3/4 = 5 \Rightarrow 5 = 5$

Property (iii):  $3 \setminus 3 = 2/2 \Rightarrow 1 = 1$

To further help with the notation, we rewrite these equations in function notation by using the left and right inverse mappings.

Property (i):  $3L(2)^{-1}L(2)$  and  $5R(3)^{-1}R(3)$

Property (ii):  $2L(4)L(4)^{-1}$  and  $5R(4)R(4)^{-1}$

Property (iii):  $3L(3)^{-1}$  and  $2R(2)^{-1}$

By Definition 2.6 we have that every loop is a quasigroup. Also, by Theorem 2.1, we have that if  $(G, \cdot)$  is a loop, every element  $a \in (G, \cdot)$  has a unique local left and right inverse such that  $bL(b)^{-1} = e$ , and  $bR(b)^{-1} = e$ . Note, however, that  $L(a)^{-1}$  does not imply  $L(a^\lambda)$ , nor does  $R(a)^{-1}$  imply  $R(a^\rho)$ . That is to say, loops need not contain unique elements which satisfy the identities  $a^\lambda \cdot (a \cdot x) = x$  and  $(x \cdot a) \cdot a^\rho = x$ . We show this in the following example.

**Example 2.3**

$\cdot$	1	2	3	4	5
1	1	2	3	4	5
2	2	4	5	1	3
3	3	1	2	5	4
4	4	5	1	3	2
5	5	3	4	2	1

In this example we have that  $2 \cdot 4 = 1$ , however  $(4 \cdot 2) \cdot 4 = 5 \cdot 4 \neq 4$ . We will see that loops which satisfy this property are unique, and carry many interesting properties which are discussed in the following section.

### 2.1.1 Inverses

As mention previously in the last section, this section will focus mainly on the restrictions needed for a loop to satisfy the identities  $a^\lambda \cdot (a \cdot x) = x$  and  $(x \cdot a) \cdot a^\rho = x$ , which we will call inverse property loops. We then go on to discuss some of the defining properties of these loops. We will then end this section by giving some additional inverse properties which loops may also possess. To begin this section we define the mappings needed for these inverse property loops, starting in quasigroups.

**Definition 2.5** *Let  $(G, \cdot)$  be a quasigroup. If there exists a bijection  $J_\lambda : a \rightarrow a^\lambda$  on  $G$  such that  $a^\lambda \cdot (a \cdot x) = x$  for every  $x \in G$  then  $G$  is a left inverse property (L.I.P) quasigroup.*

**Definition 2.6** *Let  $(G, \cdot)$  be a quasigroup. If there exists a bijection  $J_\rho : a \rightarrow a^\rho$  on  $G$  such that  $(x \cdot a) \cdot a^\rho = x$  for every  $x \in G$  then  $G$  is a right inverse property (R.I.P) quasigroup.*

With these two mappings in place, we give the following definition of interest for the section.

**Definition 2.7** *Let  $(G, \cdot)$  be a quasigroup. If  $(G, \cdot)$  satisfies both the L.I.P and a R.I.P. then  $(G, \cdot)$  is said to be an inverse property (I.P.) quasigroup.*

We now give some of the basic properties of I.P. quasigroups.

**Theorem 2.3** *let  $(G, \cdot)$  be an I.P. quasigroup, then the following hold:*

- (i)  $J_\lambda^2 = J_\rho^2 = \iota$ , where  $\iota$  is the identity mapping (i.e.  $(a^\lambda)^\lambda = a$ , and  $(a^\rho)^\rho = a$ )
- (ii) *The equations  $a \cdot x = b$  and  $y \cdot a = b$  have solutions  $x = a^\lambda \cdot b$  and  $y = b \cdot a^\rho$  respectively.*

$$(iii) (a \cdot b)^\lambda = b^\rho \cdot a^\rho \text{ and } (a \cdot b)^\rho = b^\lambda \cdot a^\lambda.$$

$$(iv) R(a)^{-1} = R(a^\rho) \text{ and } L(a)^{-1} = L(a^\lambda).$$

$$(v) J_\lambda R(a) J_\rho = L(a^\lambda)$$

$$J_\rho R(a) J_\lambda = L(a^\rho)$$

$$J_\lambda L(a) J_\rho = R(a^\lambda)$$

$$J_\rho L(a) J_\lambda = R(a^\rho)$$

Proof:

(i) Note that  $(a^\lambda)^\lambda(a^\lambda \cdot ax) = (a^\lambda)^\lambda x$ . We also have that, by Definition 2.5,  $(a^\lambda)^\lambda(a^\lambda \cdot ax) = ax$ . Thus  $(a^\lambda)^\lambda(a^\lambda \cdot ax) = (a^\lambda)^\lambda x = ax = (a^\lambda)^\lambda(a^\lambda \cdot ax)$ . Therefore, by the right cancellation property of quasigroups,  $(a^\lambda)^\lambda = a$ . Similarly  $(a^\rho)^\rho = a$

(ii) Let  $ax = b$ . By multiplying by  $a^\lambda$  on the left of both sides we have  $a^\lambda(a \cdot x) = a^\lambda b$ . Thus, by Definition 2.5, we have  $x = a^\lambda b$ . Similarly  $y \cdot a = b$  has the solution  $y = b \cdot a^\rho$ .

(iii) Let  $ab = c$ . We then have  $a = cb^\rho$ ,  $c^\lambda a = b^\rho$ ,  $c^\lambda = b^\rho a^\rho$ . Thus we have that  $(a \cdot b)^\lambda = b^\rho \cdot a^\rho$ . A similar argument can be made to show that  $(a \cdot b)^\rho = b^\lambda \cdot a^\lambda$ .

(iv) Let  $L(a) : x \rightarrow y$ , then  $L(a)^{-1} : y \rightarrow x$ . This means that  $a \cdot x = y$ . Since  $(G, \cdot)$  is an I.P. loop, we have that  $x = a^\lambda \cdot y$ . Thus  $L(a)^{-1} = L(a^\lambda)$ .

(v) Consider  $x J_\lambda R(a) J_\rho = (x^\lambda \cdot a)^\rho = a^\lambda \cdot (x^\lambda)^\lambda = a^\lambda \cdot x = x L(a^\lambda)$ .

Consider  $x J_\rho R(a) J_\lambda = (x^\rho \cdot a)^\lambda = a^\rho \cdot (x^\rho)^\rho = a^\rho \cdot x = x L(a^\rho)$ .

Consider  $x J_\lambda L(a) J_\rho = (a \cdot x^\lambda)^\rho = (x^\lambda)^\lambda \cdot a^\lambda = x \cdot a^\lambda = x R(a^\lambda)$ .

Consider  $x J_\rho L(a) J_\lambda = (a \cdot x^\rho)^\lambda = (x^\rho)^\rho \cdot a^\rho = x \cdot a^\rho = x R(a^\rho)$ . ■

We now look into the loop case.

**Theorem 2.4** *If  $(G, \cdot)$  is an L.I.P. or an R.I.P. loop then  $J_\lambda = J_\rho = J$  (i.e.  $a^\lambda = a^\rho = a^{-1}$  where  $a \cdot a^{-1} = a^{-1} \cdot a = e$ ).*

Proof: Let  $e$  be the identity element of an L.I.P. loop  $(G, \cdot)$ . Then  $a \cdot a^\rho = e$ ,  $a^\lambda \cdot a = e$ ,  $a^\lambda(a \cdot a^\rho) = a^\rho$  and  $a^\lambda \cdot e = a^\lambda$ , thus  $a^\lambda = a^\rho = a^{-1}$ . Similarly we can show that if we have a R.I.P. loop  $(G, \cdot)$ , then  $a^\rho = e \cdot a^\rho = (a^\lambda \cdot a)a^\rho = a^\lambda$ . ■<sup>1</sup>

It is worth noting that associativity (definition 2.16) need not be assumed for a quasigroups, or a loop to have the inverse property. We make a point to show this in the following example of an I.P. loop.

**Example 2.4**

$\cdot$	1	2	3	4	5	6	7	$x$	$x^{-1}$
1	1	2	3	4	5	6	7	1	1
2	2	3	1	6	7	5	4	2	3
3	3	1	2	7	6	4	5	3	2
4	4	7	6	5	1	2	3	4	5
5	5	6	7	1	4	3	2	5	4
6	6	4	5	3	2	7	1	6	7
7	7	5	4	2	3	1	6	7	6

2

Notice that associativity fails in this example when we consider the instance  $(2 \cdot 2) \cdot 4 = 3 \cdot 4 = 7$ , while  $2 \cdot (2 \cdot 4) = 2 \cdot 6 = 5$ . We also wish to highlight the inverse property of the loop by showing three instances of it.

$$4 \cdot (5 \cdot 6) = 4 \cdot 4 = 6$$

$$4 \cdot (5 \cdot 3) = 4 \cdot 7 = 3$$

$$4 \cdot (5 \cdot 2) = 4 \cdot 6 = 2.$$

---

1. Hala O. Pflugfelder, *Quasigroups and Loops Introduction*, ed. Prof. Dr. M. Hsek Prof. Dr. B. Banaschewski Prof. Dr. H. Herrlich, vol. 7, Sigma Series in Pure Mathematics, 3-88538-007-2 (Heldermann Verlag, 1990).

2. John Slaney and Asif Ali, "Generating loops with the inverse property.," *JProc. of ESARM*, 2008, 55–66.

As mentioned before, there are some classes of loops which lack the inverse property yet have properties which are variations of the inverse property.

**Definition 2.8** A loop  $(G, \cdot)$  with identity element  $e$  is called a cross inverse property (C.I.P.) loop, if any two elements  $x, y \in L$  satisfy the relation

$$(x \cdot y) \cdot x^\rho = y$$

We now derive two additional properties of C.I.P. loops in the following lemma.

**Lemma 2.1** Let  $(L, \cdot)$  be a C.I.P. loop, then the following hold

$$(i) (xy)^\rho = x^\rho y^\rho$$

$$(ii) x \cdot yx^\rho = y$$

Proof:

(i) Let  $xy = z$ . From this we have  $(xy) \cdot x^\rho = zx^\rho$ ,  $y = zx^\rho \cdot z^\rho$ ,  $yz^\rho = x^\rho$ ,  $yz^\rho \cdot y^\rho = x^\rho y^\rho$ ,  $z^\rho = x^\rho y^\rho$ . Thus  $(xy)^\rho = x^\rho y^\rho$ .

(ii) Consider the expression  $xy \cdot x^\rho = y$ , then  $x^\rho = y \cdot (xy)^\rho$ . But  $(L, \cdot)$  is a C.I.P. loop, so we have that  $yx^\rho \cdot y^\rho = x^\rho = y \cdot x^\rho y^\rho$ . Thus,  $x \cdot yx^\rho = y$  ■

Here we give one such example of a C.I.P. loop constructed by R. Artzy, who discovered them.

**Example 2.5**

$\cdot$	1	2	3	4	5	$x$	$x^{-1}$
1	1	2	3	4	5	1	1
2	2	1	4	5	3	2	2
3	3	5	1	2	4	3	3
4	4	3	5	1	2	4	4
5	5	4	2	3	1	5	5

The loop in example 2.5 satisfies  $x = x^\rho = x^\lambda$  for every  $x \in L$ , but  $(G, \cdot)$  does not have the I.P. This can be seen when we consider  $(4 \cdot 3) \cdot 3 = 5 \cdot 3 \neq 4$ , while  $(3 \cdot 4) \cdot 3 = 2 \cdot 3 = 4$ .

It is not the case for C.I.P. loops that  $x = x^\rho = x^\lambda$  must be true for every  $x \in L$ . Instead "cycles of inverses" can be formed in these loops. This means  $x_1, x_2, \dots, x_n$  such that  $(x_k)^{rho} = x_{k+1}$ ,  $k + 1$  taken modulo  $n$ , where  $n$  is called the length of the cycle. We show this in the following example.

**Example 2.6**

·	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	3	1	7	6	9	8	5	4
3	3	1	2	5	8	7	4	9	6
4	4	7	9	8	1	2	6	3	5
5	5	4	8	6	9	1	3	7	2
6	6	9	5	3	7	4	1	2	8
7	7	6	4	9	2	8	5	1	3
8	8	5	7	2	4	3	9	6	1
9	9	8	6	1	3	5	2	4	7

In this loop the cycles are  $\{1\}$ ,  $\{2,3\}$ , and  $\{4, 5, 6, 7, 8, 9\}$ .

We end this section off by giving a more general instance of the C.I.P. loops, the weak inverse property loop (W.I.P. loop), which were introduced by J. M. Osborn.

---

3. Pflugfelder, *Quasigroups and Loops Introduction*.

4. Pflugfelder.



**Definition 2.9** A loop  $(L, \cdot)$  with the identity element  $e$  is called a weak inverse property loop if it satisfies the identical relation

$$y(xy)^\rho = x^\rho$$

**Theorem 2.5** Every C.I.P. loop has W.I.P.

Proof: Let  $(L, \cdot)$  be a C.I.P. loop. Then  $(L, \cdot)$  satisfies  $(xy) \cdot x^\rho = y$  and  $(xy)^\rho$ . Thus  $y(xy)^\rho = y(x^\rho y^\rho) = x^\rho$ , which gives  $y(xy)^\rho = x^\rho$ . ■<sup>5</sup>

### 2.1.2 subloops

In this section we explore the substructures of loops, in specific subloops. We first start by defining what it means to be a subloop.

**Definition 2.10** A non-empty subset  $H$  of a set  $G$  is a subloop (subquasigroup) of a loop (quasigroup)  $(G, \cdot)$  means that  $(H, \cdot)$  is a loop (quasigroup).

We mentioned in section 2.1 that if  $(G, \cdot)$  is a quasigroup, it followed that  $(G, \backslash)$  and  $(G, /)$  are both quasigroups. with this fact in mind we give the following theorem.

**Theorem 2.6** Let  $(G, \cdot)$  be a loop. Then a non-empty subset  $H$  of  $G$  is a subloop of the loop  $(G, \cdot)$  if and only if  $(H, \cdot)$ ,  $(H, /)$ , and  $(H, \backslash)$  are groupoids.

Proof: Assume  $(H, \cdot)$ ,  $(H, \backslash)$ , and  $(H, /)$  are all groupoids. Let  $a, b \in H$ . Since  $a, b \in G$  there exists a unique  $x \in G$  such that  $a \cdot x = b$ . This implies that  $x = a \backslash b$ . So  $x \in H$  since  $a, b \in H$ , and  $(H, \backslash)$  is a groupoid. We have that  $x$  is the only element in  $H$  such that  $a \cdot x = b$  due to  $(G, \cdot)$  being a quasigroup. Similarly, since  $(H, /)$  is a groupoid, we have that there is a unique  $y \in H$  such that  $y \cdot a = b$ . Thus,  $(H, \cdot)$  is a quasigroup meaning that  $H$  is a subquasigroup of  $(G, \cdot)$ . ■<sup>6</sup>

---

5. Pflugfelder, *Quasigroups and Loops Introduction*.

6. Pflugfelder.

One may now have some questions in regards to these substructures. Are there any properties in relation between a loop and its subloops? Are there any shared relations between the subloops of a given loop? We answer these questions in the finite case, seeing as it is the more interesting of the two cases. We start to answer these questions by first understanding what a coset is.

Let  $(G, \cdot)$  be a loop and let  $H$  be a subloop of  $G$ . If  $a \in G$ , then  $aH$  and  $Ha$  are defined by

$$aH = \{a \cdot h | h \in H\}, \text{ and } Ha = \{h \cdot a | h \in H\}$$

where both  $aH$ , and  $Ha$  are subsets of  $G$ . From this we define what it is to be a coset of  $H$ .

**Definition 2.11** *Let  $(G, \cdot)$  be a loop, let  $H$  be a subloop of  $G$ , and let  $K$  be a subset of  $G$ . Then  $K$  is a left (right) coset module  $H$  means that  $K = aH$  ( $K = Ha$ ) for some  $a \in G$ .*

Which gives us

**Definition 2.12** *Let  $(G, \cdot)$  be a loop, let  $H$  be a subloop of  $G$ . Then  $(G, \cdot)$  has a left (right) coset module  $H$  means that the set  $P$  of all left (right) cosets modulo  $H$  is a partition of  $G$ .*

Recall from set theory that if  $P$  is a partition of a non-empty set  $G$ , then  $P$  has the following properties

- (i)  $P \subseteq 2^G$
- (ii)  $X \neq \emptyset$  whenever  $X \in P$
- (iii)  $G = \cup_{X \in P} X$
- (iv)  $X = Y$  whenever  $X \in P, Y \in P$  and  $X \cap Y \neq \emptyset$

With these two definitions in place we are ready to introduce the following important theorem.

**Theorem 2.7** *Let  $(G, \cdot)$  be a loop and let  $H$  be a subloop of  $G$ . Then  $(G, \cdot)$  has a left (right) coset decomposition modulo  $H$  if and only if  $(a \cdot h)H = aH$  ( $H(h \cdot a) = Ha$ ) for all  $a \in G$  and all  $h \in H$*

Proof: Let  $e$  denote the identity element of the loop  $(G, \cdot)$  and let  $P$  be the set of all left cosets modulo  $H$ .

( $\Rightarrow$ ) Assume that  $(G, \cdot)$  has a left coset decomposition modulo  $H$ . Then  $P$  is a partition of  $G$ . Notice that for  $a \in G$  and  $h \in H$  we have that  $a \cdot h = (a \cdot h) \cdot e$ . This means that  $a \cdot h \in aH \cap (a \cdot h)H$ . Thus  $aH \in P$ ,  $(a \cdot h)H \in P$ , and  $(a \cdot h)H \cap aH \neq \emptyset$ . Since  $P$  is a partition of  $G$  we have  $(a \cdot h)H = aH$ .

( $\Leftarrow$ ) Assume that  $(a \cdot h)H = aH$  for all  $a \in G$  and all  $h \in H$ . We have  $P \subseteq 2^G$ , and for each  $g \in G$  note that  $g = g \cdot e \in gH$ . Thus  $G = \bigcup_{X \in P} X$ . Also note that  $X \in P \Rightarrow X = gH$  for some  $g \in G$  implies that  $g = g \cdot e \in gH$  meaning that  $X \neq \emptyset$ . Lastly we show that  $aH = bH$  for all  $aH$  and  $bH$  in  $P$ , where  $aH \cap bH \neq \emptyset$ . If  $aH \cap bH \neq \emptyset$  there exists a  $g \in aH \cap bH$ . Thus  $g = a \cdot x = b \cdot y$  for some  $x, y \in H$ . But, given our assumption, we have that  $aH = (a \cdot x)H = (b \cdot y)H = bH$ . Therefore  $P$  is a partition of  $G$ .

A similar argument can be used to show that  $(G, \cdot)$  has a right coset decomposition modulo  $H$  if and only if  $H(h \cdot a) = Ha$ . ■<sup>7</sup>

We now define those subloops whose order divides the order of the loop. This property was originally studied in groups first by Joseph-Louis Lagrange, where he proved that the order of the subgroup divides the order of the group in the finite

---

<sup>7</sup> Pflugfelder, *Quasigroups and Loops Introduction*.

case. We go through this proof in detail later in Section 2.2, only mentioning this fact so as not to confuse the reader as to why we call loops with this property Lagrange-like. Since, as we will see, all groups are associative loops.

**Definition 2.13** *Let  $(G, \cdot)$  be a finite loop, and let  $H$  be a subloop of  $G$  where  $|H|$  divides  $|G|$ . Then the subloop  $H$  of  $(G, \cdot)$  is said to be Lagrange-like.*

**Definition 2.14** *Let  $(G, \cdot)$  be a finite loop, then  $(G, \cdot)$  satisfies the weak lagrange property meand that every subloop of  $(G, \cdot)$  is lagrange-like*

**Definition 2.15** *Let  $(G, \cdot)$  be a finite loop. Then  $(G, \cdot)$  satisfies the strong lagrange property means that  $(H, \cdot)$  satisfies the weak lagrange property whenever  $H$  is a subloop of  $(G, \cdot)$ .*

Cosets, defined before in Definition 2.11, are what give us the relations we were looking for between loops and their subloops. We show this in the following theorem.

**Theorem 2.8** *Let  $(G, \cdot)$  be a finite loop and let  $H$  be a subloop of  $G$ . If  $(G, \cdot)$  has a left (right) coset decomposition modulo  $H$  then  $H$  is Lagrange-like*

Proof: Let  $(G, \cdot)$  have a left coset decomposition modulo  $H$  and let  $P$  be the set of all left cosets modulo  $H$ . Since  $P$  is a partition we have

$$|G| = \sum_{x \in P} |X|.$$

Let  $X \in P$ . We now note that  $X = aH$  for some  $a \in G$ . By defining  $\alpha$  to be  $h\alpha = a \cdot h$  for all  $h \in H$  it is clear that  $\alpha : H \rightarrow aH$  is a bijection since  $(G, \cdot)$  is a quasigroup. Hence,  $|H| = |X|$ . It follows then that

$$|G| = \sum_{x \in P} |X| = m|H|$$

where  $m = |P|$ . Thus,  $|H|$  divides  $|G|$ . A similar argument is made in the case

when  $G$  has a right coset decomposition modulo  $|H|$ . ■<sup>8</sup>

A consequence of Theorems 2.7, and 2.8 is the following.

**Theorem 2.9** *Let  $(G, \cdot)$  be a finite loop and let  $H$  be a subloop of  $G$ . If  $(a \cdot h)H = aH$  ( $H(h \cdot a) = Ha$ ) for all  $a \in G$  and all  $h \in H$ , then  $H$  is Lagrange-like. ■<sup>9</sup>*

### 2.1.3 Moufang Loops

In this section we look at one of the most well known loops, Moufang loops, named after Ruth Moufang. We start with the original definition given by Moufang, who called them quasigroups, then give their current definition. We go onto giving some of their basic properties, and end the section with the commutative Moufang loops.

A quasigroups  $Q^*$ , as defined by Moufang, was said to be a set with multiplication such that the following hold

( $M_1$ ) for any two elements  $x, y$  there exist a unique product  $xy$ ;

( $M_2$ ) there exists an identity element 1, and to any element  $x$ , there is a unique  $x^{-1}$  such that  $x^{-1}x = 1 = xx^{-1}$ ;

( $M_3$ ) for any  $x$  and  $y$ :  $x(x^{-1}y) = (xx^{-1})y$  and  $(yx^{-1})x = y(x^{-1}x)$ ;

( $M_4$ ) for any  $x, y, z$ :  $[x(zx)]y = x[z(xy)]$ .

A quasigroup  $Q^{**}$  additionally satisfies

( $M_5$ )  $(xy)(zx) = x[(yz)x]$ .

Moufang went on to show that ( $M_4$ ) was equivalent to

---

8. Pflugfelder, *Quasigroups and Loops Introduction*.

9. Pflugfelder.

$$(M_6) [(xz)x]y = x[z(xy)]$$

and

$$(M_7) [(yx)z]x = y[x(zx)].$$

We now give our current definition of a Moufang loop, due to G. Bol, and R. H. Bruce who proved the equivalence of the identities  $(M_4) - (M_7)$ , now called the Moufang identities.

**Definition 2.16** *A loop  $(M, \cdot)$  with identity element 1 is called a Moufang loop if it satisfies the Moufang identity*

$$(MI) \quad (xy)(zx) = [x(yz)]x.$$

We now give the smallest example of a non-associative Moufang loop.

**Example 2.7**

·	1	2	3	4	5	6	7	8	9	a	b	c
1	1	2	3	4	5	6	7	8	9	a	b	c
2	2	1	4	3	6	5	8	7	c	b	a	9
3	3	6	5	2	1	4	9	a	b	c	7	8
4	4	5	6	1	2	3	a	9	8	7	c	b
5	5	4	1	6	3	2	b	c	7	8	9	a
6	6	3	2	5	4	1	c	b	a	9	8	7
7	7	8	b	a	9	c	1	2	5	4	3	6
8	8	7	c	9	a	b	2	1	4	5	6	3
9	9	c	7	8	b	a	3	4	1	6	5	2
a	a	b	8	7	c	9	4	3	6	1	2	5
b	b	a	9	c	7	8	5	6	3	2	1	4
c	c	9	a	b	8	7	6	5	2	3	4	1

10

To show the Moufang property, we work through the following multiplications.

$$(23)(92) = 4 \cdot c = b = a \cdot 2 = (2 \cdot b)2 = [2(39)]2$$

$$(4a)(c4) = 7 \cdot b = 3 = 2 \cdot 4 = (4 \cdot 5)4 = [4(ac)]4$$

$$(ab)(ca) = 2 \cdot 3 = 4 = 7 \cdot a = (a \cdot 4)a = [a(bc)]a.$$

We also show that associativity fails when considering the following multiplication.

$$(2 \cdot 3) \cdot 7 = 4 \cdot 7 = a \neq c = 2 \cdot 9 = 2 \cdot (3 \cdot 7).$$

We now prove and some additional properties of all Moufang loops which follow from Definition 2.16. We go onto show that all the Moufang identities are equivalent in

---

10. Petr Vojtechovsky Michael Kinyon Kyle Pula, “Incidence Properties of Cosets in Loops.,” *J. Combinatorial Designs* 20 (2012): 161–197, <https://doi.org/10.1002/cod.1108>.

the following theorem.

**Theorem 2.10** *A Moufang loop is an I.P. loop and satisfies the identical relations  $(xx)y = x(xy)$   $x(yy) = (xy)y$ , and  $(xy)x = x(yx)$ . Furthermore, all Moufang identities are equivalent.*

Proof: Here we prove all 9 statements in the most convenient order.

(i)  $y^\lambda(yx) = x$  where  $y^\lambda y = 1$ , and  $y^\lambda = y^\rho = y^{-1}$ . (L.I.P.)

Let  $y^\lambda y = 1$ , and substitute  $y^\lambda$  in for  $x$  in  $(MI)$ , thus we have that  $(zy^\lambda z) = [y^\lambda(yz)]y^\lambda$ . This implies that  $y^\lambda(yz) = z$ .

(ii)  $(xy)x = x(yx)$  (flexible law)

By setting  $y = 1$  in  $(MI)$ , we have that  $x(zx) = (xz)x$ .

(iii)  $(M_5)$  and  $(M_I)$  are equivalent.

Trivial application of  $(ii)$ .

(iv)  $(xy)y^{-1} = x$  where  $yy^{-1} = 1$  (R.I.P.)

Let  $z = x^{-1}$  in  $(M5)$  to give:

$$xy = (xy)(x^{-1}x) = x[(yx^{-1})x], \text{ or } (yx^{-1}x) = y.$$

(Therefore we have that  $(M, \cdot)$  is an I.P. loop.)

(v)  $(M_7)$  and  $(M_I)$  are equivalent.

We refer the reader to the orange book for this proof

(vi)  $(M_6)$  and  $(M_7)$  are equivalent.

These are inverses of one another.

(vii)  $(xx)y = x(xy)$  (left alternative law).

By letting  $z = 1$  in  $(M_6)$ , we have that  $y(yx) = (yy)x$ .

(viii)  $(xy)y = x(yy)$

This is derived from  $(vii)$



(ix)  $(M_4)$  and  $(MI)$  are equivalent.

A trivial application of (ii). ■<sup>11</sup>

We now give the definition for a commutative loop.

**Definition 2.17** *A loop  $(L, \cdot)$  is commutative means that  $L(a) = R(a)$  for all  $a \in G$*

With Definition 2.17 and the Moufang identity, we end this section by defining a Commutative Moufang loop.

**Definition 2.18** *A loop  $(M, \cdot)$  with identity element 1 is called a commutative Moufang loop if it satisfies the relation*

$$x^2(yz) = (xy)(xz).$$

## 2.2 Groups

We begin this section by giving the definition of a group. We then go on to prove some basic properties of groups.

**Definition 2.19** *A groupoid  $(G, \cdot)$  is associative means that  $R(a \cdot b) = R(a)R(b)$  for all  $a, b \in G$*

**Definition 2.20** *A groupoid  $(G, \cdot)$  is a group means that  $(G, \cdot)$  is an associative quasigroup.*

In the case of loops, one must assume an identity element. This, however, is not the case for groups. An associative quasigroup necessarily has a unique identity element, which we show in the following theorem.

**Theorem 2.11** *If  $(G, \cdot)$  is a quasigroup which is associative, then  $(G, \cdot)$  necessarily has a unique identity element.*

---

<sup>11</sup>. Pflugfelder, *Quasigroups and Loops Introduction*.

Proof: Since  $G$  is non-empty, there is an element  $a$  in  $G$ . Now by Theorem 2.1 there is  $e \in G$  so that  $a \cdot e = a$ . Let  $b$  be any element in  $G$ . Again by Theorem 2.1 there is a  $y \in G$  so that  $y \cdot a = b$ . It follows that  $bR(e) = b \cdot e = (y \cdot a)e = yR(a)R(e) = yR(a \cdot e) = yR(a) = y \cdot a = b$ . Thus,  $R(e) : G \rightarrow G$  is the identity map on  $G$ , and so  $e$  is the right identity element for  $(G, \cdot)$ .

Now let  $b$  be any element in  $G$ . We have  $b \cdot b = (b \cdot e) \cdot b = bR(e)R(b) = bR(e \cdot b)$ . By the left cancellation we note that  $b \cdot b = b \cdot (e \cdot b)$  implies  $b = e \cdot b$ . This is true for all  $b \in G$ , so  $bL(e) = b$  for all  $b \in G$ . But  $L(e) : G \rightarrow G$  being the identity map on  $G$  means that  $e$  is a left identity element for  $(G, \cdot)$ . Thus, we know that  $e$  is an identity element for  $(G, \cdot)$  and is, in fact, the only one. ■<sup>12</sup>

In section 2.2 we saw some of the many types of inverse properties which can take place in loops. This is due to associativity not being assumed. In groups, however, we do not have this problem. All groups satisfy the property  $a^\lambda \cdot (a \cdot x) = x$  and  $(x \cdot a) \cdot a^\rho = x$ . Thus they are all inverse property loops, which we prove in the following theorem.

**Theorem 2.12** *If  $(G, \cdot)$  is a group, then it satisfies the inverse property.*

Proof: Since  $G$  is nonempty, there is an element  $a \in G$ . Now by Theorem 2.1 there is a  $a^\lambda \in G$  such that  $a^\lambda \cdot a = e$ . Since  $G$  is associative we have  $a^\lambda \cdot (a \cdot x) = (a^\lambda \cdot a) \cdot x = x$ , which is always true by Theorem 2.1. The other proof follows similarly. ■

We now give a theorem that lays most of the ground work for groups. The following theorem is more of a corollary of the previous theorems found in Section 2.1. We, however, choose to write new proofs to highlight the use of associativity which greatly simplify those previous proofs.

**Theorem 2.13** *If  $(G, \cdot)$  is a group, then*

---

<sup>12</sup>. Pflugfelder, *Quasigroups and Loops Introduction*.

(i) every  $a \in G$  has a unique inverse in  $G$ :

(ii) the right and left cancellation laws hold

(iii) for every  $a \in G$ ,  $(a^{-1})^{-1} = a$

(iv) for all  $a, b \in G$ ,  $(a \cdot b)^{-1} = b^{-1} \cdot a^{-1}$ .

Proof:

(i) Since  $G$  is not empty, there exists an element  $a \in G$ . Let  $b \cdot a = e$  and  $a \cdot c = e$ .

Using associativity we then have that

$$b = b \cdot e = b \cdot (a \cdot c) = (b \cdot a) \cdot c = e \cdot c = c.$$

Thus every element  $a \in G$  has a unique inverse.

(ii) Assume that  $a \cdot x = a \cdot y = e$  for  $a, x, y$  in  $G$ . There exists an element  $b \in G$  such that  $b \cdot a = e$ . Thus  $b \cdot (a \cdot x) = b \cdot (a \cdot y)$ . By the associative law we have

$$x = e \cdot x = (b \cdot a) \cdot x = b \cdot (a \cdot x) = b \cdot (a \cdot y) = (b \cdot a) \cdot y = e \cdot y = y$$

Similarly we can prove that  $x \cdot a = y \cdot a$  implies that  $x = y$ . Thus the right and left cancellation laws hold for groups.

(iii) Consider the equation  $a^{-1} \cdot (a^{-1})^{-1} = e = a^{-1} \cdot a$ . Cancellation on the left side by  $a^{-1}$  gives us  $(a^{-1})^{-1} = a$ .

(iv) Consider the equation  $(a \cdot b) \cdot (b^{-1} \cdot a^{-1}) = a \cdot ((b \cdot b^{-1}) \cdot a) = a \cdot (e \cdot a^{-1}) = a \cdot a^{-1} = e$ .

Thus  $(a \cdot b)^{-1} = b^{-1} \cdot a^{-1}$ . ■<sup>13</sup>

---

13. I. N. Herstein, *Topics in Algebra*, 63-17982 (Blaisdell Publishing Company, 1964).

### 2.2.1 Subgroups

In this section we outline what it means to be a subgroup. We then look at their properties. We start by giving the definition of a group.

**Definition 2.21** *A subset  $H$  of a group  $G$  is said to be a subgroup of  $G$  if, under the product in  $G$ ,  $H$  itself forms a group.*

From this definition we see that if  $H$  is a subgroup of  $G$  and  $K$  is a subgroup of  $H$ , then  $K$  is also a subgroup of  $G$ . We now need a method for determining when a chosen subset of a group is a subgroup, which we give in the following theorem.

**Theorem 2.14** *A nonempty subset  $H$  of the group  $G$  is a subgroup of  $G$  if and only if*

(i)  $a, b \in H$  implies that  $a \cdot b \in H$

(ii)  $a \in H$  implies that  $a^{-1} \in H$

Proof: If  $H$  is a subgroup of  $G$ , then (i), (ii) must hold.

Suppose conversely that  $H$  is a subset of  $G$  for which (i) and (ii) hold. In order to establish that  $H$  is a subgroup all that is needed is to verify that  $e \in H$  and that the associative law holds for elements in  $H$ . Since the associative law does hold for  $G$ , it holds all the more so for  $H$  which is a subset of  $G$ . If  $a \in H$ , by (ii)  $a^{-1} \in H$  and so by (i)  $e = a \cdot a^{-1} \in H$ . ■<sup>14</sup>

We have previously seen what it means to be a right or left coset in loops. In particular we saw that a given subloop needn't be a right or left coset of a loop. Theorem 2.9 makes it clear what requirements are needed to guarantee that a given subloop is a left or right coset. We will use Theorem 2.9 to prove that every subgroup of a group is a left or right coset, thus every subgroup is Lagrange-like.

---

<sup>14</sup> Herstein, *Topics in Algebra*.

From this we can show that any groups satisfies the strong Lagrange property.

Before we prove that, we first define a coset of a group.

**Definition 2.22** *If  $H$  is a subgroup of  $G$ ,  $a \in G$ , then  $Ha = \{ha|h \in H\}$  ( $aH = \{ah|h \in H\}$ ).  $Ha$  ( $aH$ ) is called a right (left) coset of  $H$  in  $G$ .*

**Theorem 2.15** *Let  $(G, \cdot)$  be a group, and let  $H$  be a subgroup of  $G$ . Then  $(G, \cdot)$  satisfies the strong Lagrange property.*

Proof: Given theorem 2.9, we need only prove that  $(a \cdot h)H = aH$  for all subgroups.

$$(a \cdot h)H = a \cdot (h \cdot H) = aH \blacksquare$$

It is worth noting that without the help of theorem 2.9, the original proof took a very different shape. So, in the spirit of Lagrange, we outline how the proof goes. To do this, however, we must give some preliminary definitions, theorems, and lemmas.

**Definition 2.23** *The binary relation,  $\sim$ , on  $A$  is said to be an equivalence relation on  $A$  if for all  $a, b, c$  in  $A$ :*

(i)  $a \sim a$

(ii)  $a \sim b$  implies  $b \sim a$

(iii)  $a \sim b$  and  $b \sim c$  implies  $a \sim c$

**Definition 2.24** *If  $A$  is a set and if  $\sim$  is an equivalence relation on  $A$ , then the equivalence class of  $a \in A$  is the set  $\{x \in A|a \sim x\}$ . We write it as  $cl(a)$ .*

**Theorem 2.16** *The distinct equivalence classes of an equivalence relation on  $A$  provide us with a decomposition of  $A$  as a union of mutually disjoint subsets. Conversely, given a decomposition of  $A$  as a union of mutually disjoint, nonempty subsets, we can define an equivalence relation on  $A$  for which these subsets are distinct equivalence classes.<sup>15</sup>*

---

<sup>15</sup>. Herstein, *Topics in Algebra*.

**Definition 2.25** Let  $G$  be a group,  $H$  a subgroup of  $G$ ; for  $a, b \in G$  we say  $a$  is congruent to  $b$  mod  $H$ , written as  $a \equiv b \pmod{H}$  if  $ab^{-1} \in H$

**Lemma 2.2** The relation  $a \equiv b \pmod{H}$  if  $ab^{-1} \in H$  is an equivalence relation.<sup>16</sup>

With these previous statements above, we now give the to following two lemmas which prove Lagrange's Theorem.

**Lemma 2.3** For all  $a \in G$ ,  $Ha = \{x \in G \mid a \equiv x \pmod{H}\}$ .<sup>17</sup>

**Lemma 2.4** there is a 1-1 correspondence between any two right cosets of  $H$  in  $G$ .<sup>18</sup>

Thus we have

**Theorem 2.17** If  $G$  is a finite group and  $H$  is a subgroup of  $G$ , then the order of  $H$  divides the order of  $G$  (Lagrange's Theorem).<sup>19</sup>

---

16. Herstein, *Topics in Algebra*.

17. Herstein.

18. Herstein.

19. Herstein.

## 3 Computability Theory

We start this section by first giving an informal sense of what it means to be computable. Say you are handed a machine that, when given an input  $x$ , yields an output after finitely many steps. The machine may yield on all, or only on a given subset of inputs. A machine which yields an output on all of its given inputs is called computable. To make this generalization precise, we give two mathematical formulations of computability. We will see that these, and many other such definitions, both describe the same class of functions which are called the computable functions.

### 3.1 Formal Definitions of Computable Functions

It has been shown that the two formal definitions we will introduce, partial recursive and Turing machines, give rise to exactly the same class called the computable functions. We give these two definitions to show that there are many intuitive ways to think about computability, where any one can be used based on the individuals preference. This is due to Church's Thesis which asserts that these functions, along with the many others, coincide with the intuitively computable functions. We will assume Church's Thesis in this paper and will use the terms "partial recursive (p.r)," "Turing computable," and "computable" interchangeably.

#### 3.1.1 Turing Machines

In this section we give a formal definition of a Turing machine, created by Alan Turing. A Turing machine,  $M$ , contains a two-way infinite tape, a reading head, and a finite set of internal states. The tape of the Turing machine is divided into cells similar to that of a film strip. The reading head scans one cell of the tape at a time, and we denote the set of internal states as  $Q = \{q_0, q_1, q_2, \dots, q_n\}, n \geq 1$ . The cells of the tape are either blank (B) or have the symbol 1 written on them. The machine may simultaneously: (1) change from one state to another; (2) change

the scanned symbol  $s$  to another symbol  $s' \in S = \{1, B\}$ ; and (3) move the reading head one cell to the right (R) or left (L) in a single step. The operation of  $M$  is controlled by a partial map  $\delta : Q \times S \rightarrow Q \times S \times \{R, L\}$  (which may not be defined for all arguments). You can break this down as, if  $(q, s, q', s', X) \in \delta$  then the machine  $M$  in state  $q$ , scanning symbol  $s$  changes to state  $q'$ , replaces  $s$  by  $s'$ , and moves to scan one cell to the right if  $X = R$  (left if  $X = L$ ). The map  $\delta$ , viewed as a finite set of quintuples, is called a Turing program. A given integer input  $x$  is represented by a string of  $x+1$  consecutive 1's (with all the other cells blank). To help with breaking down this definition we give a diagram, and run through an example of a Turing machine computing, as well as its program, for a given function.

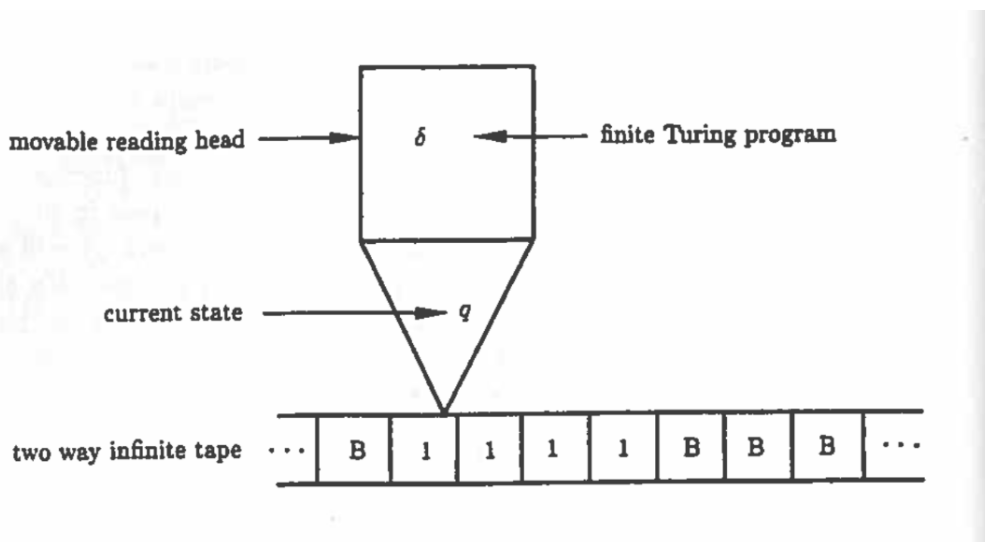


Figure 1: Turing Machine

20

Here we lay out how a Turing machine would compute the function  $f(x) = x + 4$ .  $M$  begins in the starting state,  $q_1$ , scanning the left-most cell containing a 1. The starting state, in this example, tells the reading head (1) if you read a 1 stay in  $q_1$

---

20. Robert I. Soare, *Recursively Enumerable Sets and Degrees*, 978-3-540-66681-3 (Springer-Verlag Berlin Heidelberg New York, 1987).



replace the 1 with a 1 and move right, or (2) if you read a blank move to state  $q_2$  and replace the blank with a 1 and move right. In  $q_2$  the reading head need only worry about the case where a blank is read, since the input  $x$  starts as a consecutive string of 1's. So  $q_2$  says if you read a blank move to state  $q_3$  and replace the blank with a 1. In  $q_3$  the reading head reads a blank moves to the halting state  $q_0$  and replace the blank with a 1. Once in in the halting state we have that, without loss of generality, no further moves take place. Another way to think of these states is as commands where  $q_1$  is the command "find first blank and replace it with a 1,"  $q_2$  "change second blank to 1" and state  $q_3$  is the command "replace next blank with one." The output is then the total number of 1's on the tape. Writing this in Turing machine code gives:

$$\begin{array}{ccccc}
 q_1 & 1 & q_1 & 1 & R \\
 q_1 & B & q_2 & 1 & R \\
 q_2 & B & q_3 & 1 & R \\
 q_3 & B & q_0 & 1 & R
 \end{array}$$

In general, if the halting state  $q_0$  is reached by  $M$ , we say  $M$  halts and the output  $y$  is the total number of 1's on the tape.  $M$  computes the partial function  $\psi$  provided that  $\psi(x) = y$  if and only if  $M$  with input  $x$  eventually halts and yields output  $y$ .

The sequence of configurations  $c_0, c_1, \dots, c_n$  of a Turing program  $P$  with input  $x$  are called the Turing computation. The machine in starting state  $q_1$  reading the leftmost symbol of the input  $x$  is denoted by  $c_0$ . The halting state  $q_0$  of the Turing machine is denoted by  $c_n$ . The transitions  $c_i \rightarrow c_{i+1}$ , for all  $i < n$ , is given by the Turing program  $P$ . A partial function of  $n$  variables is associated with with each Turing machine  $M$  by representing the input  $(x_1, x_2, \dots, x_n)$  by the following initial configuration of  $M$ ,  $q_1\alpha_1B\alpha_2\dots B\alpha_n$  where  $\alpha_i$  consists of  $x_i + 1$  consecutive 1's.

### 3.1.2 Primitive Recursive Functions

In this section we give another formal definition of the computable functions called the partial recursive function. We start off by defining the primitive recursive functions, but will see that the primitive recursive functions, even though they contain all the computable functions from number theory, fail to contain all computable functions. To allow for all computable functions we add an additional schemata onto the definition of the primitive recursive functions to give the partial recursive function of Kleene.

**Definition 3.1** *The class of primitive recursive functions is the smallest class  $C$  of functions closed under the following schemata.*

(I) *The successor function,  $\lambda x[x + 1]$ , is in  $C$ .*

(II) *The constant functions,  $\lambda x_1 \dots x_n[m]$ , is in  $C$ ,  $0 \leq n, m$ .*

(III) *The identity functions,  $\lambda x_1 \dots x_n[x_i]$ , is in  $C$ ,  $1 \leq n, 1 \leq i \leq n$ .*

(IV) *(Composition) if  $g_1, g_2, \dots, g_m, h \in C$ , then*

$$f(x_1, \dots, x_n) = h(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n))$$

*is in  $C$  where  $g_1, \dots, g_m$  are functions of  $n$  variables and  $h$  is a function of  $m$  variables.*

(V) *(Primitive Recursion), If  $g, h \in C$  and  $n \geq 1$  then  $f \in C$  where*

$$\begin{aligned} f(0, x_2, \dots, x_n) &= g(x_2, \dots, x_n) \\ f(x_1 + 1, x_2, \dots, x_n) &= h(x_1, f(x_1, x_2, \dots, x_n), x_2, \dots, x_n) \end{aligned}$$

*assuming  $g$  and  $h$  are functions of  $n - 1$  and  $n + 1$  variables respectively.*

A primitive recursive function is called a derivation, or a sequence,  $f_1, f_2, \dots, f_k = f$  such that each  $f_i$ ,  $i \leq k$  is either an initial function ((I), (II), or (III)), or  $f_i$  is obtained from  $\{f_j : j < i\}$  by (IV) or (V). We give an example of this with the function  $f(x_1, x_2) = x_1 + x_2$ . and run through program with the inputs  $f(3, 2)$ .

**Example 3.1** *The derivation of the function  $f(x_1, x_2) = x_1 + x_2$ .*

$$f_1 = \lambda x[x + 1]$$

$$f_2 = \lambda x[x]$$

$$f_3 = \lambda x_1 x_2 x_3[x_2]$$

$$f_4 = \lambda f_1 \circ f_3$$

$$f_5(0, x_2) = f_2(x_2)$$

$$f_5(x_1 + 1, x_2) = f_4(x_1, f_5(x_1, x_2), x_2)^{21}$$

We now work through the specific instance of  $f(3, 2)$  to give a better understanding of schemata (V).

$$f(3, 2) = f_4(2, f_5(2, 2), 2)$$

$$f(2, 2) = f_4(1, f_5(1, 2), 2)$$

$$f(1, 2) = f_4(0, f_5(0, 2), 2) = f_4(0, 2, 2) = 3$$

$$f(2, 2) = f_4(1, f_5(1, 2), 2) = f_4(1, 3, 2) = 4$$

$$f(3, 2) = f_4(2, f_5(2, 2), 2) = f(3, 2) = f_4(2, 4, 2) = 5.$$

As we mentioned before, the primitive recursive functions include all of the usual functions from elementary number theory, yet they fail to give all computable functions. We make this clear with the following theorem.

**Theorem 3.1** *The primitive recursive functions fail to include all computable functions.*

---

21. Soare, *Recursively Enumerable Sets and Degrees*.

Proof: Given that each derivation of a primitive recursive function is a finite string of symbols from a fixed alphabet, all derivations can be listed. Let  $f_n$  be the functions corresponding to the  $n$ th derivation in this listing. Now consider the function  $g(x) = f_x(x) + 1$ . This function cannot be primitive recursive since  $g \neq f_n$  for all  $x$ , yet  $g_n$  is computable. ■<sup>22</sup>

Seeing that  $g_n$  is computable on the other hand is not so obvious. To help aid in seeing this fact note that to compute  $g_n$  we call the  $n$ th primitive recursive program, give it input  $n$  and add 1 to the output. Thus  $g_n$  is computable. Similar diagonalization arguments can be used when applying any effective set of schemata which produce only total functions. So, to avoid this and obtain all computable functions, we use computable partial functions. These are those functions which may not be defined on all arguments. Diagonalization arguments are no longer a worry when considering partial functions. For example, let  $\psi_n$  be the partial function computed by the  $n$ th algorithm under some effective coding of all algorithms. Suppose  $\varphi(x) = \psi_x(x) + 1$  if  $\psi_x(x)$  is defined and  $\varphi(x)$  is defined otherwise. Now if  $\varphi$  corresponds to the  $x_0$ th algorithm then diagonalization does not imply that  $\varphi \neq \psi_{x_0}$  since  $\psi_{x_0}(x_0)$  may be undefined. So With this in mind we give our next definition.

**Definition 3.2** *The class of partial recursive (p.r.) functions is the least class obtained by closing under schemata (I) through (V) for the primitive recursive functions and the following schemata (VI). A total recursive function (abbreviated recursive function) is a partial recursive function which is total.*

(VI) (Unbounded Search) *If  $\theta(x_1, \dots, x_n, y)$  is a partial recursive function of  $n + 1$  variables, and*

$$\psi(x_1, \dots, x_n) = \mu y [\theta(x_1, \dots, x_n, y) \downarrow = 0 \ \& \ (\forall z \leq y) [\theta(x_1, \dots, x_n, z) \downarrow]]$$

*then  $\psi$  is a partial recursive function of  $n$  variables.*

---

<sup>22</sup>. Soare, *Recursively Enumerable Sets and Degrees*.

## 3.2 Computability Background

In the last section we gave two different definitions of computability and assumed, by Church's Thesis, that these definitions coincided with one another. In this section we will go through some of the basic results, notations, and definitions of computability theory. We omit the formal proofs of Theorems 3.2, and 3.4 given by Kleene and Hermes. We instead give a general outline of how to prove them.

### 3.2.1 Basic Results

**Notation 3.1** We let  $\langle x, y \rangle$  denote the image of  $(x, y)$  under the standard pairing function  $\beta_2(x, y) = \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y)$ . This computable function is a bijection that takes  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ . Let  $\pi_1$  and  $\pi_2$  denote the inverse functions  $\pi_1(\langle x, y \rangle) = x$ , and  $\pi_2(\langle x, y \rangle) = y$ . Let  $\langle x_1, x_2, x_3 \rangle$  denote  $\langle \langle x_1, x_2 \rangle, x_3 \rangle$ , and  $\langle x_1, x_2, x_3, \dots, x_n \rangle$  denote  $\langle \dots \langle \langle x_1, x_2 \rangle, x_3 \rangle, \dots, x_n \rangle$ .

**Definition 3.3** A relation  $R \subseteq \mathbb{N}^n$ ,  $n \geq 1$ , is recursive (primitive recursive, has property  $P$ ) if its characteristic function  $\chi_R$  is recursive, (respectively primitive recursive, has property  $P$ ) where  $\chi_R(x_1, \dots, x_n) = 1$  if  $(x_1, \dots, x_n) \in R$  and  $= 0$  otherwise. Note that a set  $A \subseteq \mathbb{N}$  corresponds to the case  $n = 1$  so we have the definition of a set being recursive.

One such computable relation would be  $R = \{x \mid x \text{ is even}\}$ , since both the even and odd numbers are primitive recursive. We give a stronger primitive recursive derivation for both the even and odd numbers in the following example, but simply substituting 2 in for  $x_1$  yields the wanted result.

**Example 3.2** Derivation of the function  $f(x_1, x_2) = (x_1 \times x_2)$ .

$$f_1 = \lambda x[x] + 1$$

$$f_2 = \lambda x[x]$$

$$f_3 = \lambda x_1 x_2 x_3[x_2]$$

$$\begin{aligned}
f_4 &= f_1 \circ f_3 \\
f_5(0, x_2) &= f_2(x_2) \\
f_5(x_1 + 1, x_2) &= f_4(x_1, f_5(x_1, x_2), x_2) \\
f_6(x_1, x_2, x_3) &= f_5(x_1, x_3) \\
f_7(x_1, 0) &= 0 \\
f_7(x_1, x_2 + 1) &= f_6(x_1, x_2, f_7(x_1, x_2)).
\end{aligned}$$

**Example 3.3** *Derivation of the function  $f(x_1, x_2) = (x_1 \times x_2) + 1$ .*

$$\begin{aligned}
f_1 &= \lambda x[x] + 1 \\
f_2 &= \lambda x[x] \\
f_3 &= \lambda x_1 x_2 x_3[x_2] \\
f_4 &= f_1 \circ f_3 \\
f_5(0, x_2) &= f_2(x_2) \\
f_5(x_1 + 1, x_2) &= f_4(x_1, f_5(x_1, x_2), x_2) \\
f_6(x_1, x_2, x_3) &= f_5(x_1, x_3) \\
f_7(x_1, 0) &= 0 \\
f_7(x_1, x_2 + 1) &= f_6(x_1, x_2, f_7(x_1, x_2)). \\
f_8 &= f_1 \circ f_7
\end{aligned}$$

The relation  $R = \{x \mid x \text{ is prime}\}$  can also be shown to be computable. If we let  $p_0, p_1, \dots$  denote the primes in increasing order we have that for any  $x \in \mathbb{N}$   $x = p_0^{x_0} p_1^{x_1} \dots p_n^{x_n} \dots$ , is unique when finitely many  $x_i \neq 0$ . The function

$$(x)_i = \text{the exponent } x_i \text{ of } p_i$$

from the previous equation can also be shown to be computable. Thus any finite sequence  $\{a_0, a_1, \dots, a_n\}$  of positive integers has a unique code number  $a = p_0^{a_0+1} \dots p_n^{a_n+1}$  such that  $a_i = (a_i)$  can be obtained primitive recursively from  $a$ . With this one can show that a code number can be assigned to each Turing program and config-

uration, since that every Turing program is a finite set of quintuples. These code numbers are known as a Gödel number. With this we give the following definition.

**Definition 3.4** *Let  $P_e$  be the Turing program with code number  $e$  (also called index  $e$ ) in this listing and let  $\varphi_e^{(n)}$  be the partial function of  $n$  variables computed by  $P_e$ , where  $\varphi_e$  abbreviates to  $\varphi_e^{(1)}$ .*

**Lemma 3.1** *Each partial recursive function  $\varphi_x$  has  $\aleph_0$  indices, furthermore for each  $x$  we can effectively find an infinite set  $A_x$  of indices for the same partial function.*

Proof: For any program  $P_x$  with internal states  $\{q_0, \dots, q_n\}$ , we can add the additional instructions  $q_{n+1}B q_{n+1}B R, q_{n+2}B q_{n+2}B R, \dots$ , to give a new program for the same function. ■<sup>23</sup>

**Theorem 3.2** *There exists a predicate  $T(e, x, y)$  and a function  $U(y)$  which are recursive such that*

$$\varphi(x) = U(\mu y T(e, x, y)).$$

Sketch of proof: Informally, the predicate asserts that  $y$  is the code number of some Turing computation according to the program  $P_e$  with input  $x$ . To check that  $T(e, x, y)$  holds, we recover from  $e$  the program  $P_e$ . We then recover from  $y$  the computation  $c_0, c_1, \dots, c_n$  if  $y$  codes such a computation. Next we check whether  $c_0, c_1, c_n$  is a computation according to  $P_e$  with  $x$  as the input in  $c_0$ . If  $x$  is,  $U(y)$  simply outputs the number of 1's in the final configuration  $c_n$ . To prove that both  $T$  and  $U$  are primitive recursive, use the unique coding of Turing programs and computations. ■<sup>24</sup>

We now introduce two theorems which are foundational to computability theory.

---

23. Soare, *Recursively Enumerable Sets and Degrees*.

24. Soare.

**Theorem 3.3** For every  $n \geq 1$  there exists a partial recursive function  $\varphi_{z_n}(e, x_1, \dots, x_n)$  of  $n+1$  variables such that  $\varphi_{z_n}(e, x_1, \dots, x_n) = \varphi_e^{(n)}(x_1, \dots, x_n)$  for all  $e$  and  $x_1, \dots, x_n$ .

Proof: By Theorem 3.2 let  $\varphi_{z_n}(e, x_1, \dots, x_n) = U(\mu y T(e, x_1, \dots, x_n, y))$ . ■<sup>25</sup>

**Theorem 3.4** For every  $m, n \geq 1$  there exists an injective recursive function  $s_n^m$  of  $m+1$  variables such that for all  $x, y_1, y_2, \dots, y_m$

$$\varphi_{s_n^m(x, y_1, y_2, \dots, y_m)}^{(n)} = \lambda z_1, \dots, z_n [\varphi_x^{m+n}(y_1, \dots, y_m, z_1, \dots, z_n)].$$

Informal Proof: Consider the case  $m = n = 1$ . The program  $P_{s_1^1}$  on input  $z$  first obtains  $P_x$  and then applies  $P_x$  to input  $(y, z)$ . By Church's Thesis we have that  $s = s_1^1$  is recursive since this is an effective procedure in  $x$  and  $y$ . If we have that  $s$  is not already injective we may replace it with an injective recursive function  $s'$  by Lemma 3.1 to give  $\varphi_{s(x, y)} = \varphi_{s'(x, y)}$ . This is done by defining  $s'(x, y)$  in increasing order of  $\langle x, y \rangle$ , where  $\langle x, y \rangle$  is the image of  $(x, y)$  under the pairing function notation 3.1. ■<sup>26</sup>

### 3.2.2 Recursively Enumerable Sets

We end this section of computability theory with some of the interesting paradoxes and dilemmas that come about when studying the natural numbers. The question of answering if a set or function is computable, computably enumerable, and or non-computable is where much of computability theory lies. We start this section by giving the standard definition of a computably enumerable set. We then end this section with defining a computably enumerable set which is not computable.

**Definition 3.5** (i) A set  $A$  is computably enumerable (c.e.) if  $A$  is the domain of some partial recursive function.

(ii) Let the  $e$ th c.e. set be denoted by

---

25. Soare, *Recursively Enumerable Sets and Degrees*.

26. Soare.



$$W_e = \text{dom } \varphi_e = \{x \mid \varphi_e(x) \downarrow\} = \{x \mid (\exists y) T(e, x, y)\}.$$

The existence of these c.e. sets, as we shall see in section 4, have been seen in other various areas of mathematics. We look at the studies involving these unsolvable problems in the case of the algebraic structures of groups and loops. Here we give one such unsolvable set we call  $K$ , and look at some of its properties.

**Definition 3.6** Let  $K = \{x \mid \varphi_x(x) \text{ converges}\} = \{x \mid x \in W_x\}$ .

**Proposition 3.1**  $K$  is c.e.

Proof:  $K$  is the domain of the following partial recursive function

$$\psi(x) = \begin{cases} x & \text{if } \varphi_x(x) \text{ converges} \\ \text{undefined} & \text{otherwise} \end{cases} \quad \blacksquare^{27}$$

One could also prove this using Theorem 3.3 by noting that,  $K = \text{dom } \theta$  where  $\theta(x) = \varphi_z^2(x, x)$ .

**Proposition 3.2**  $K$  is not computable.

Proof: Assume, for contradiction, that  $K$  had a computable characteristic function  $\chi_K$ . The function

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

Would then be computable. But  $f \neq \varphi_x$  for any  $x$ , thus a contradiction.  $\blacksquare^{28}$

---

27. Soare, *Recursively Enumerable Sets and Degrees*.

28. Soare.

## 4 Applied Computability

One area of mathematics where computability has been applied to is the structure of a group. This was due to the question posed by Dehn in 1911, known as the word problem for groups. This problem was independently proven to be unsolvable by both Boone and Novicov if the group was finitely generated. Instead of working through these challenging proofs, we instead give some different examples of computable and noncomputable groups. We do, however, prove later that finitely presented abelian groups have solvable word problem.

### 4.1 Computable Groups

**Definition 4.1** *A computable group  $G$  consist of a set  $\{a_0, a_1, a_2, \dots\}$ , indexed by elements of  $\mathbb{N}$  or by a finite subset of  $\mathbb{N}$ , with binary operation  $(\cdot)$  on these elements such that:*

(i)  $\{a_0, a_1, a_2, \dots\}$  forms a group under these operations, and

(ii) *There exists computable functions  $f, g, h: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  such that for all elements  $a_i$  and  $a_j$ ,  $a_i \cdot a_j = a_{f(i,j)}$*

From definition 4.1, we see that the other basic operations on these structures are also computable. The identity element can be found in a computable group by computing  $f(0,0), f(1,1), \dots$ , since the identity element  $a_i$  has the unique index  $i$  satisfying  $f(i,i) = i$ . With the identity, one can then compute the inverse function (the function  $v$  such that  $a_n^{-1} = a_{v(n)}$ ), by computing  $f(n,0), f(n,1), \dots$  until the unique  $m$  with  $f(n,m) = i$  is found. We only concern ourselves with finitely generated, and later finitely presented structures, since computability theory is defined on the set of natural numbers  $\mathbb{N}$ .

We now give our first example of a computable group using the free (nonabelian)  $\mathbf{FG}_\omega$  on countably many generators. First we will index the domain of  $G$ ,  $\{a_0, a_1, a_2, \dots\}$ ,

using the bijection  $\beta : \mathbb{N}^* \rightarrow \mathbb{N}$  defined previously, and also the bijection  $\gamma : \mathbb{N} \rightarrow \mathbb{Z}$  with  $\gamma(k) = (-1)^k \cdot \lfloor \frac{k+1}{2} \rfloor$ . We let  $z_0, z_1, z_2, \dots$  be the generators, and let  $\beta(k_0, \dots, k_n) = i$ . This gives us that a given group element  $a_i$  represents the word

$$z_{|\gamma(k_0)|}^{(-1)^{k_0}} \dots z_{|\gamma(k_n)|}^{(-1)^{k_n}}.$$

This gives us that the generators are  $a_{\beta(0)}, a_{\beta(2)}, a_{\beta(4)}, \dots$ , with  $a_{\beta(2m+1)} = a_{\beta(2m)}^{-1}$ .

The group multiplication on  $G$  is given by concatenation

$$a_{\beta(j_0, \dots, j_m)} \cdot a_{\beta(k_0, \dots, k_n)} = a_{\beta(j_0, \dots, j_m, k_0, k_n)},$$

where if  $k_0 = j_m + (-1)^{j_m}$ , the middle terms cancel each other out and similarly for the following middle terms. This gives us the computable group  $G$  isomorphic to the free group on the generators  $a_{\beta(0)}, a_{\beta(2)}, a_{\beta(4)}, \dots$ .<sup>29</sup>

A surprising fact is that there are also many noncomputable group representations with domain  $\{a_0, a_1, \dots\}$  isomorphic to the  $FG_\omega$ . To see this we use an arbitrary bijection  $p$  taking  $\mathbb{N}$  onto  $\mathbb{N}$ . Notice that this forms a group under the following multiplication:

$$a_i \cdot a_j = a_{p^{-1}(f(p(i), p(j)))}$$

where  $f$  is the same computable function used in the previous example. Since only countably many of the continuum many bijections  $p$  can be computable, there must exist many noncomputable representations of  $FG_\omega$  on the domain  $\{a_0, a_1, \dots\}$ .<sup>30</sup>

There are also more concrete way of showing that a group is noncomputable. Let  $H$  be a group defined in a similar way to the computable representation of  $FG_\omega$ ,

---

<sup>29</sup>. Russell Miller, *An Introduction to Computable Model Theory on Groups and Fields*, 2011, <https://qcpages.qc.cuny.edu/~rmiller/DK.pdf>.

<sup>30</sup>. Miller.

only changing the definition so as to make the generators  $a_{\beta(0)}$  and  $a_{\beta(2n)}$  commute with each other iff  $n$  lies in the c.e. set  $K$ . We can see that  $H$  is not computable by considering the function  $g$  which would compute it. If the group  $H$  was solvable, then the function  $g$  that satisfies the logic statement

$$(\forall n \in \mathbb{N})[n \in K \iff g(0, 2n) = g(2n, 0)]$$

would be solvable. However, deciding membership in  $K$  is impossible.<sup>31</sup>

It may now come as no surprise to hear that there exists a computable representation isomorphic to the group  $H$ . Let  $J$  be defined similarly except for the generators  $a_{\beta(0)}$  and  $a_{\beta(2n)}$  commute with each other iff  $n$  lies in the set of prime numbers  $P$ . We previously stated before that the set of prime numbers is computable. If we let  $k$  be the function on indices defined by its multiplication, then

$$(\exists n \in \mathbb{N})[n \neq P \iff k(0, 2n) = k(2n, 0)],$$

which is a computable statement. Hence  $k$  is a computable function, thus  $J$  is a computable group.

## 4.2 Computable Loops

We can define a computable loop in much the same way as we defined a computable group in Definition 4.1. However, it sometimes proves useful to deal with the words of a loop, defined by its generators and relations, instead to prove whether the loop is computable or noncomputable. Normal forms do exactly that since they solve for uniquely determined words which represent equivalence classes of equal words (e.g. reduced words in free groups). This is why normal forms are used as a way

---

<sup>31</sup>. Miller, *An Introduction to Computable Model Theory on Groups and Fields*.

to determine the the solvability or unsolvability of the word problem for loops, and other algebras. Normal forms, however, can require a great deal of ingenuity. We give a normal form theorem by Evans in 1950 to help those unfamiliar with to such theorems an example. The normal form we give, however, does not prove the solvability or unsolvability of a loop. The normal form theorem we give shows that for a loop defined by a closed set of relations there exists a unique word of shortest length in each equivalence class which can be derived in a simple manner from any other word in the class. We omit the proof of the normal form theorem as it is rather long. However, we give the preliminary ideas which are needed to solve the theorem. We then give an additional theorem and two corollaries which allow us to state some properties of normal forms.

**Definition 4.2** *We define a word in a set of generators  $g_1, g_2, g_3, \dots$  by*

*(i)  $g_1, g_2, g_3, \dots$  are words;*

*(ii) if  $u$ , and  $v$  are words, then so is  $u \circ v$*

*The words from which a given word is built up are called its components. That is,*

*(i) the only components of a generator is the generator itself;*

*(ii) the only components of a word  $w = u \circ v$  are the word itself and the components of  $u$ , and  $v$  (called the major components).*

The length of the word  $w$  is denoted as  $l(w)$ , and is defined as  $l(u \circ v) = l(u) + l(v)$ . The length of a generator is taken as 1. We let  $r_i(g_1, g_2, g_3, \dots) = r'_i(g_1, g_2, g_3, \dots)$  ( $i = 1, 2, \dots$ ) be the set of equations between words in  $g_1, g_2, g_3, \dots$ . Two words in  $(g_1, g_2, g_3, \dots)$  are said to be equivalent if we can transform one word into the other by a finite sequence of applications of the loop axioms and the equations  $r_i = r'_i$ . The resulting equivalence classes form a loop if one defines  $\{u\} \circ \{v\}$  to be  $\{u \circ v\}$ , where  $\{w\}$  denotes the equivalence class containing the word  $w$ . This loop is called the loop generated by  $g_1, g_2, g_3, \dots$  with relations  $r_i(g_1, g_2, g_3, \dots) = r'_i(g_1, g_2, g_3, \dots)$ .

We let the unit element of this loop be the equivalence class  $\{a \setminus a\}$ . We assume, that the unit element  $e$ , from now on for convenience, is included among the set of generators.

**Definition 4.3** *Let  $L$  be a loop generated by  $g_1, g_2, g_3, \dots$ . We say that its relations are in closed form if they satisfy the following conditions:*

- (i) *everyone of the relations is of the form  $x \circ y = z$ , where  $x, y$ , and  $z$  are generators;*
- (iia) *if  $x \cdot y = z$  is a relation, then so are  $x \setminus z = y$  and  $z / y = x$ ;*
- (iib) *if  $x \setminus y = z$ , then so are  $x \cdot z = y$  and  $y / z = x$ ;*
- (iic) *if  $x / y$  is a relation, then so are  $z \cdot y = x$  and  $z \setminus x = y$ ;*
- (iii) *no two relations occur such as  $x \cdot y = z$ ,  $x \cdot y = z'$ , identifying two generators  $z, z'$ .*
- (iv) *For all  $x$ ,  $x \cdot e = x$ ,  $e \cdot x = x$ , are included in the set of relations.*

**Definition 4.4** *Let  $L$  be a loop defined by a closed set of relations. Let  $w$  be a word in the generators  $e, g_1, g_2, g_3, \dots$  of  $L$ . By an elementary reduction of  $w$ , we mean the replacing of a component of  $w$  of the form*

- (ia)  $u \setminus u$  by  $e$ ;
- (ib)  $u / u$  by  $e$ ;
- (iia)  $e \cdot u$  by  $u$ ;
- (iib)  $u \cdot e$  by  $u$ ;
- (iia)  $e \setminus u$  by  $u$ ;
- (iib)  $u / e$  by  $u$ ;
- (iva)  $u \cdot (u \setminus v)$  by  $v$ ;
- (ivb)  $(v / u) \cdot u$  by  $v$ ;
- (va)  $u \setminus (u \cdot v)$  by  $v$ ;
- (vb)  $(v \cdot u) / u$  by  $v$ ;
- (via)  $u / (v \setminus u)$  by  $v$ ;
- (vib)  $(u / v) \setminus u$  by  $v$ ;

where  $u, v$  are words,

(vii)  $x \circ y$  by  $z$ ,

if  $x \circ y = z$  is one of the relations of the defining relations of  $L$ .

The opposite process to any one of these elementary reductions is called an elementary expansion. These elementary reductions and expansions are the applications of the loop's axioms and relations. We say that two words in the generators of  $L$  represent the same element of  $L$  if, and only if a finite sequence of transformation can be given from one word to the other via the elementary reductions and expansions. A word is said to be a normal form in the generators of a loop  $L$  if no elementary reductions of the word is possible.

**Notation 4.1** We denote a single elementary reduction from a word  $u$  to a word  $v$  by  $u \rightarrow v$ . A sequence of word reduction, denoted by  $w_1 \rightarrow w_2 \rightarrow \dots \rightarrow w_k$ , is called a reduction chain.

We now give the following normal form theorem. We then give an additional theorem and two corollaries that follow from the normal form theorem.

**Theorem 4.1** Let  $w$  be any word in  $L$ . If  $w \rightarrow w_1 \rightarrow \dots \rightarrow w_p$  and  $w \rightarrow w'_1 \rightarrow \dots \rightarrow w'_q$  are two reduction chains such that  $w_p, w'_q$  are the same.<sup>32</sup>

**Theorem 4.2** If  $u$  and  $v$  are two equivalent words, then the reduction chains  $u \rightarrow u_1 \rightarrow \dots$  and  $v \rightarrow v_1 \rightarrow \dots$  end in the same normal form<sup>33</sup>

**Corollary 4.1** Two normal forms are equivalent if, and only if, they are identical.<sup>34</sup>

**Corollary 4.2** A normal form is the shortest word in the equivalence class containing it.<sup>35</sup>

---

32. Trevor Evans, "On Multiplicative Systems Defined by Generators and Relations: I. Normal Form Theorems.," *Mathematical Proceedings of the Cambridge Philosophical Society* 47, no. 4 (1951): 673–649, <https://doi.org/10.1017/S0305004100027092>.

33. Evans.

34. Evans.

35. Evans.

From the previous theorems and corollaries we are able to give some additional properties of normal forms. If  $w$  is a normal form then so is every component of  $w$ . Conversely, if  $u, v$  are two normal forms then the word  $u \circ v$  is either normal or else there is only one reduction possible which necessarily involves both major components. The normal forms are the actual elements of the loop  $L$ , with the operations defined in the obvious way. The relations between the length of the normal form  $w$ , equivalent to  $u \circ v$ , and the lengths  $l(u), l(v)$  are

- (i) if  $u \circ v$  is normal,  $l(w) = l(u) + l(v)$ ,
- (ii) if  $u \circ v$  is not normal  $l(w)$  is  $|l(u) - l(v)|, l(u), l(v)$ , or 1

We start this final section by looking at computability theory applied to finitely presented algebras in general. We define an algebra.

### 4.3 Finitely Presented Algebras

In this final section we look at purely algebraic properties which allow for the construction of an algorithm to solve the word problem.

**Definition 4.5** *An algebra,  $\mathbb{A} = (A, \Omega)$ , is a nonempty set  $A$  with a finite set  $\Omega$  of finitary operations  $f : A^n \rightarrow A$ , where  $n = 1, 2, 3, \dots$ , with each  $n$ -ary operation being a mapping of  $A^n$  onto  $A$ .*

We denote the general algebra with an unspecified set of finitary operations as  $f(x_1, x_2, \dots, x_n)$  for the value of the  $n$ -ary operations  $f$  at  $(x_1, x_2, \dots, x_n)$ . Examples of some algebras we have shown are groupoids (having the one binary operation  $x \cdot y$ ), quasigroups (having the three binary operations  $x \cdot y, x/y, x \setminus$ ), and groups (having the binary operation  $x \cdot y$ , and the unary operation  $x^{-1}$ ).

**Definition 4.6** *A variety of algebras,  $\mathbb{V}$ , is the class of all algebras having a specified set of operations  $\Omega$  which satisfy a specified set of axioms each in the form of an identity.*



An algebra  $\mathbb{A}$  is in  $\mathbb{V}$  if the identities of  $\mathbb{V}$  are satisfied by the operations and elements of  $\mathbb{A}$ . Given a variety  $\mathbb{V}$ , we can describe an algebra in terms of generators  $g_1, g_2, g_3, \dots$  and relations  $r_1 = r'_1, r_2 = r'_2, r_3 = r'_3, \dots$ . An element in a  $\mathbb{V}$ -algebra  $\mathbb{A}$  is represented by expressions built from the generators  $g_1, g_2, g_3, \dots$  and the operations of  $\mathbb{V}$ . The relations in the defining identities of  $\mathbb{A}$  are words in the generators  $g_1, g_2, g_3, \dots$ . The defining identities of  $\mathbb{V}$  are equations  $u(x_1, x_2, x_3, \dots) = v(x_1, x_2, x_3, \dots)$ , where  $u$  and  $v$  are words in variables  $x_1, x_2, x_3, \dots$ . The elements of  $\mathbb{A}$  are equivalence classes of words in the generators. The words  $w(g_1, g_2, g_3, \dots)$ , and  $w'(g_1, g_2, g_3, \dots)$  are equivalent if there is a finite sequence of substitutions, using the identities and relations of  $\mathbb{A}$ , which transforms  $w$  into  $w'$  denoted by  $w \rightarrow w'$ . We depict this in the following example.

**Example 4.1** *Let  $\mathbb{V}$  be a variety of groupoids defined by the identities  $x = x^2$  and  $(xy) \cdot x = y$ , for all  $x, y$ . Let  $\mathbb{A}$  be the  $\mathbb{V}$ -algebra generated by  $a, b$  with the defining relation  $ba = ab \cdot b$ . We have*

$$((ba) \cdot ((aa)b)) \cdot (ab \cdot b) \rightarrow ((ba) \cdot ((aa)b)) \cdot (ba) \rightarrow (aa) \cdot b \rightarrow ab$$

*Thus, the word  $((ba) \cdot ((aa)b)) \cdot (ab \cdot b) = ab$  in  $\mathbb{A}$ .*

Let  $\mathbb{V}^\emptyset$  denote the the variety of all algebras of the same operation type as  $\mathbb{V}$ .  $\mathbb{V}^\emptyset$  can be thought of as having the same operations as  $\mathbb{V}$  but defined by the empty set of identities. Let  $\mathcal{F}$  be the free algebra in  $\mathbb{V}^\emptyset$  generated by  $(g_1, g_2, g_3, \dots)$ .  $\mathbb{A}$  can be regarded as the quotient algebra  $\mathcal{F}/\theta$  where  $\theta$  is the congruence of  $\mathcal{F}$  generated by all pairs  $(r_i, r'_i)$  and all instances  $(u, v)$  of the defining identities of  $\mathbb{V}$ . An effective procedure can be given for generating  $\theta$ . This is to say that, there exists an algorithm for listing all equations  $w = w'$  which hold in  $\mathbb{A}$ ; the subset of all  $(w_i, w'_i)$  such that  $w_i = w'_i$  in  $\mathbb{A}$  is a recursively enumerable subset of the set of all pairs of words in  $\mathcal{F}$ .

**Definition 4.7** *Let  $\mathbb{A}$  be a finitely presented algebra (both generators and relations are finite) in a variety  $\mathbb{V}$  such that, for any elements  $x \neq y$  in  $\mathbb{A}$ , there is a*

homomorphism  $\alpha : \mathbb{A} \rightarrow \mathbb{B}$  onto a finite algebra such that  $x\alpha \neq y\alpha$  in  $\mathbb{B}$ . Then  $\mathbb{A}$  is residually finite.

With this definition, we are able to construct an algorithm for solving the word problem for  $\mathbb{A}$ . We do this to prove the following theorem.

**Theorem 4.3** *A finitely presented residually finite algebra has a solvable word problem.*

Proof: Let  $\mathbb{A}$  be a finitely presented algebra in a finitely presented variety  $\mathbb{V}$  and let  $u(g_1, g_2, g_3, \dots), v(g_1, g_2, g_3, \dots)$  be two words in the generators of  $\mathbb{A}$ . There is an effective enumeration of all equations  $r(g_1, g_2, g_3, \dots) = s(g_1, g_2, g_3, \dots)$  which follow from the defining relations of  $\mathbb{A}$ . To see this, imagine a machine  $M_1$  that systematically lists all pairs of words  $(w, w')$  in the congruence on  $F_n(\mathbb{V}^\emptyset)$  which is generated by the defining identities of  $\mathbb{V}$  and the defining relations of  $\mathbb{A}$ . This equation will eventually be produced by  $M_1$ , if  $u = v$  holds in  $\mathbb{A}$ .

Now assume that  $\mathbb{A}$  is residually finite. We now imagine a second machine  $M_2$  which systematically constructs all finite algebras in  $\mathbb{V}$  and computes for each one all homomorphism of  $\mathbb{A}$  into it. If  $u \neq v$  in  $\mathbb{A}$ , then because of its residual finiteness of  $\mathbb{A}$ , in one of these homomorphisms the images  $u$  and  $v$  will be distinct. Combining these two machines, we see that after a finite number of steps either machine  $M_1$  stops because  $u = v$  has happened in its enumeration, or the machine  $M_2$  stops because it has found a finite homomorphic image of  $\mathbb{A}$  which separates  $u$  and  $v$ . In either case, the algorithm stops after a finite number of steps, giving an answer to the question: is  $u = v$  in  $\mathbb{A}$ ? ■<sup>36</sup>

From this we have that finitely presented abelian groups, and commutative Moufang loops have solvable word problems, since they are residually finite.

---

36. Trevor Evans, "Word problems," *Bulletin of the American Mathematical Society* 84, no. 5 (1978): 789–802, <https://doi.org/bams/1183541140>.

There still exists many other open problems that are similar in nature to the word problem.

1. The isomorphism problem: Is there an algorithm for deciding whether two finitely presented  $\mathbb{V}$ -algebras are isomorphic
2. Is there an algorithm for deciding whether a finitely presented  $\mathbb{V}$ -algebra is free?
3. Is there an algorithm for deciding, for an  $n$ -generator free *mathds* $\mathbb{V}$ -algebra, whether a set of  $n$  elements is a free generating set?

## REFERENCES

- Evans, Trevor. “On Multiplicative Systems Defined by Generators and Relations: I. Normal Form Theorems.” *Mathematical Proceedings of the Cambridge Philosophical Society* 47, no. 4 (1951): 673–649. <https://doi.org/10.1017/S0305004100027092>.
- . “Word problems.” *Bulletin of the American Mathematical Society* 84, no. 5 (1978): 789–802. <https://doi.org/bams/1183541140>.
- Herstein, I. N. *Topics in Algebra*. 63-17982. Blaisdell Publishing Company, 1964.
- Michael Kinyon, Petr Vojtechovsky, Kyle Pula. “Incidence Properties of Cosets in Loops.” *J. Combinatorial Designs* 20 (2012): 161–197. <https://doi.org/arXiv:1108.3656>.
- Miller, Russell. *An Introduction to Computable Model Theory on Groups and Fields*, 2011. <https://qcpages.qc.cuny.edu/~rmiller/DK.pdf>.
- Pflugfelder, Hala O. *Quasigroups and Loops Introduction*. Edited by Prof. Dr. M. Hsek Prof. Dr. B. Banaschewski Prof. Dr. H. Herrlich. Vol. 7. Sigma Series in Pure Mathematics, 3-88538-007-2. Heldermann Verlag, 1990.
- Slaney, John, and Asif Ali. “Generating loops with the inverse property.” *JProc. of ESARM*, 2008, 55–66.
- Soare, Robert I. *Recursively Enumerable Sets and Degrees*. 978-3-540-66681-3. Springer-Verlag Berlin Heidelberg New York, 1987.